Modelling of Surface Ships using Artificial Neural Networks

Kirkegaard, Poul Henning; Jensen, F. M.; Thoft-Christensen, Palle

Publication date:
1996

Document Version
Publisher's PDF, also known as Version of record

INSTITUTTET FOR BYGNINGSTEKNIK
DEPT. OF BUILDING TECHNOLOGY AND STRUCTURAL ENGINEERING
AALBORG UNIVERSITET • AUC • AALBORG • DANMARK

STRUCTURAL RELIABILITY THEORY
PAPER NO. 165

P. H. KIRKEGAARD, F. M. JENSEN & P. THOFT-CHRISTENSEN
MODELLING OF SURFACE SHIPS USING ARTIFICIAL NEURAL NETWORKS
JULY 1996 ISSN 1395-7953 R9625

The STRUCTURAL RELIABILITY THEORY papers are issued for early dissemination of research results from the Structural Reliability Group at the Department of Building Technology and Structural Engineering, University of Aalborg. These papers are generally submitted to scientific meetings, conferences or journals and should therefore not be widely distributed. Whenever possible reference should be given to the final publications (proceedings, journals, etc.) and not to the Structural Reliability Theory papers.

Printed at Aalborg University

INSTITUTTET FOR BYGNINGSTEKNIK
DEPT. OF BUILDING TECHNOLOGY AND STRUCTURAL ENGINEERING
AALBORG UNIVERSITET • AUC • AALBORG • DANMARK

STRUCTURAL RELIABILITY THEORY
PAPER NO. 165

P. H. KIRKEGAARD, F. M. JENSEN & P. THOFT-CHRISTENSEN
MODELLING OF SURFACE SHIPS USING ARTIFICIAL NEURAL NETWORKS
JULY 1996 ISSN 1395-7953 R9625

PREFACE

CONTENTS

1. INTRODUCTION

For various design and planning purposes there is at present an increasing interest and a need for numerical modelling of the process of navigating a vessel (or a floating body in general). The reason for this is that experiments in "full mission" simulators with human navigators at the handles are costly and time consuming - at least in the context of sampling sufficient data for statistical analysis purposes. Furthermore, a modelling of the process of navigation could be used in a vessel autopilot to guide a vessel on a fixed heading (course - keeping) or a new heading (course - changing).

Therefore, generation of data for analyses of processes involving human interaction (offshore operations, ship navigation, lifting operations, etc.) is obviously very conveniently solved if the capacity and performance of the behaviour of a human navigator can be numerically modelled. If it is possible to make a numerical model of a human navigator the effect of human errors on a design and planning process could be reduced. A human navigator model used to guide a vessel may also imply a reduction of unforeseen human errors which will increase the safety. However, in order to model a human navigator one should be able to model numerically 1) the ship and 2) how a human will navigate a ship.

In this report the first of these two problems is investigated. These investigations look into the possibility of using artificial neural networks for modelling a ship. Artificial neural networks are computational models inspired by the neuron architecture and operation of the human brain. The pioneering work in this field is usually attributed to McCulloch et al. [1]. They developed a simplified model of a neuron. The brain is composed of neurons of many different types, see e.g. McCulloch et al. [1]. For a more detailed description of neural networks, see e.g. Hertz et al. [2] and Hush et al. [3]. Since McCulloch and Pitts in 1943 there have been many studies of mathematical models of neural networks. Many different types of neural networks have been proposed by changing the network topology, node characteristics and learning procedures. Examples of those are e.g. the Hopfield network, see Hopfield [4], the Kohonen network, see Kohonen [5], and the so-called multi-layered perceptron (MLP). The MLP (also sometime called the Backpropagation neural network) is currently given the greatest attention by application developers, see e.g. Rumelhart et al. [6]. Most of the work where neural networks have been used for modelling dynamic mechanical systems has been based on multi-layered feedforward networks trained by the backpropagation algorithm, see e.g. Masri et al. [7], Qi et al. [8], Riva et al. [9]. However, such static neural networks cannot by themselves represent dynamic systems. The static neural network can only be used to model dynamic systems if the inputs and outputs to the network are selected based on the understanding of the system to be modelled. Generally, in order to model non-linear dynamic systems, it is necessary to use the so-called recurrent neural networks which incorporate feedback, see e.g. Hush et al. [3], Hertz et al. [2], Seidl et al.[10], Pham et al. [11] and Sjöberg [12]. Modelling of a ship is a non-linear dynamic problem where one should expect that recurrent networks should be used. The problem of modelling a ship using neural networks has been considered in e.g. Burns [13], Endo et al. [14], Witt et al. [15] and Balasuriya et al. [16]. In these papers the traditional MLP based static networks have been used. The objective of this report is to investigate modelling of ships using three different types of neural networks. The three different neural network approaches are described in chapter 2. In chapter 4 the three approaches are used to model a single-screw high-speed container ship in surge, sway, roll and yaw based on simulated data. The assumptions for these simulations are given in chapter 3.

2. ARTIFICIAL NEURAL NETWORKS

This chapter outlines the characteristics of the three different types of artificial neural networks used in this report for modelling a ship model of a single-screw high-speed container ship in surge, sway, roll and yaw. The three neural network types outlined in sections 2.1, 2.2 and 2.3 are the Multi-Layer Perceptron network (MLPN), Radial Basis Function network (RBFN) and Recurrent neural network (RN), respectively.

2.1 MULTI-LAYER PERCEPTRON

The Multi-Layer Perceptron network (MLPN) belongs to the class of layered feed-forward nets with supervised learning. This network is probably the most often considered member of the neural network family. The main reason for this is its ability to model simple as well as very complex functional relationships. This has been proved through a large number of practical applications, see e.g. Hertz et al. [2], Topping et al. [11].

An MLPN is made up of one or more hidden layers placed between the input and output layers, see fig. 2.1. Each layer consists of a number of nodes connected in the structure of a layered network. The typical architecture is fully interconnected, i.e. each node in a lower level is connected to every node in the higher level. Output units cannot receive signals directly from the input layer. During the training phase activation flows are only allowed in one direction, a feed-forward process, from the input layer to the output layer through the hidden layers. The input vector feeds each of the first layer nodes, the outputs of this layer feed into each of the second layer nodes and so on.

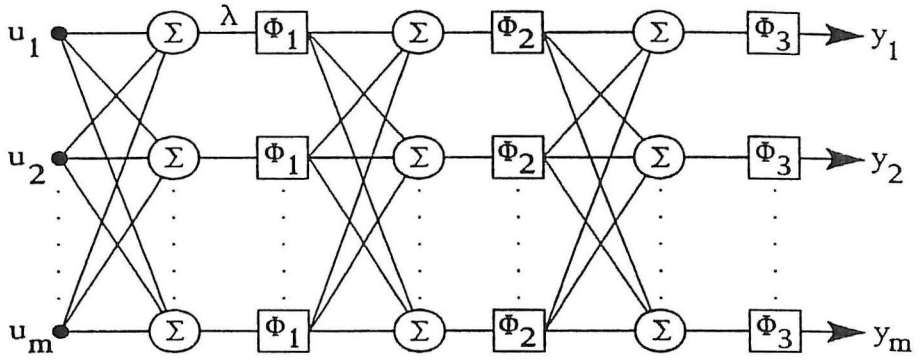


Figure 2.1: Principle of a 3-layer Multi-Layer Perceptron neural network.

Associated with each connection between node i in the preceding layer and node j in the following layer is a numerical value which is the strength or the weight of the connection. At the start of the training process these weights are initialized by small random values.
For the MLPN network with n layers in figure 2.1 signal pass through the network and the input vector $\boldsymbol{u}$ is linked to the output vector $\boldsymbol{y}$ by means of the equation

$$y = \phi_n(W_n\phi_{n-1}(W_{n-1}\ldots\phi_1(W_1\boldsymbol{u} + b_1) + \ldots.b_{n-1}) + b_n) \quad (1)$$

where W_i is the weight matrix associated with the ith layer, b_i indicate threshold or bias values associated with each node in the ith layer and ϕ_i is a non-linear operator or activation function associated with the ith layer. The function ϕ_i is assumed to be differentiable and to have a strictly positive first derivative. For the nodes in the hidden layers, the activation function is often chosen to be a so-called sigmoidal function

$$\phi(\lambda) = \frac{1}{1+e^{-\lambda}} \quad (2)$$

The activation function for the nodes in the input and output layers is often chosen as linear. In figure 2.1, a 3 layer network is shown, with m input signals and m output signals. It is however by no means necessary for the number of inputs to be equal to the number of outputs. In fact this is usually not the case. Also, it is more usual for the number of nodes in the hidden layer to be greater than those in either the input or output layer.

Important points in the selection of a suitable MLPN are a) how many layers of neurons/nodes should there be and b) how many neurons/nodes should there be in each layer. These questions relate directly to model structure problems in system identification, the aim being to choose the "best" structure to obtain a high degree of accuracy.

The first stage of creating an artificial MLPN to model an input-output system is to establish the appropriate values of the connection weights and thresholds by using a learning algorithm. A learning algorithm is a systematic procedure for adjusting the weights and thresholds in the network to achieve a desired input/output relationship, i.e. supervised learning. The most popular and successful learning algorithm used to train MLPN is currently the so-called Back-propagation routine, see e.g. Rumelhart [1]. The Back-propagation algorithm normally employs a gradient descent search technique for minimizing an error, normally defined as the mean square difference between desired and estimated outputs of output node for all the training data. The thresholds are adjusted in the same way as the weights. The process of computing the gradient and adjusting the weights and thresholds is repeated until a minimum of the error (or a point sufficiently close to the minimum) is found. The mean square error is not known a priori but must be constructed from the known errors at the output layer. The errors are passed backwards through the net and a training algorithm uses the error to adjust the connection weights moving backwards from the output layer, layer by layer, hence the name "Back-propagation".In the traditional implementation of the Back-propagation training algorithm the gradients are multiplied by a so-called "learning rate" which is chosen as large as possible without leading to intolerable oscillations. To overcome this problem, a momentum term is usually also introduced into the updating rule. However, it is generally true that the convergence of the back-propagation algorithm is fairly slow. Attempts to speed the learning include variations on simple gradient search, line search techniques and second order techniques, see e.g. Hertz et al. [2], Billings et al. [18]. In this report the training rule is the traditional gradient descent search technique implemented in the `trainbpx` m-file used in the Matlab Neural Networks Toolbox, Demuth et al. [19].

2.2 Radial Basis Function

Recently Radial Basis Function Networks (RBFN) have been shown to operate well in a neural network context and indeed it has been shown, in theory, that they can model any arbitrary nonlinear system, see Chen et al. [20].
The general structure of an RBFN is much simple than the MLPN, in particular the output layer is merely a linear combination of the signals from one hidden layer. The basic structure of an RBFN is shown in figure 2.2. The transfer function of the output layer is linear. The transfer function for the nodes in the hidden layer has to be a function, which produce a significant non zero response only when the input falls within a small localized region of the input space.

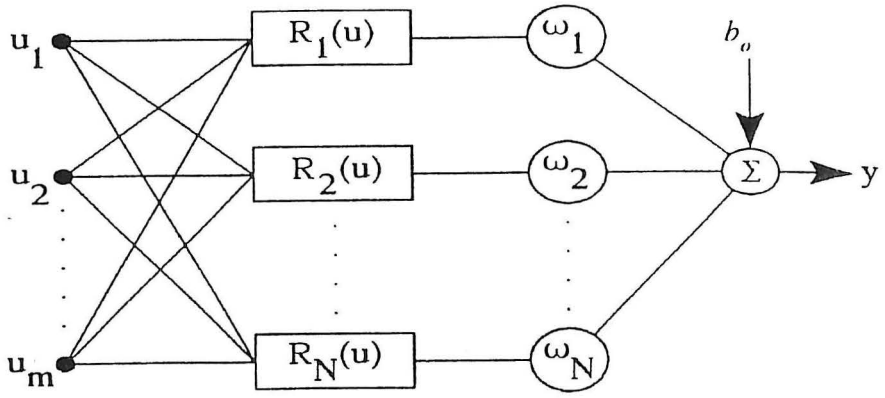


Figure 2.2: Principle of a Radial Basis Function neural network.

For the RBFN in figure 2.2, the single output signal, y, is found, in relation to the m-element input vector, u, by means of the equation

$$y = \sum_{n=1}^{N} w_i H(u) + b_0 \qquad (3)$$

where w_i (i=1,...,N) are the network weights and b_0 is the bias term. The basis functions $H_i(u)$ are then given by the expression

$$H_i(u) = \phi(\|u - c_i\|) \qquad (4)$$

in which the basis function centres c_i (i=1,...N) are each m-vectors. Thus the input to a radial basis neuron is the vector distance between its weight vector and the input vector multiplied by the weight. The basis function $\phi(\bullet)$ itself can be selected in one of a number of forms. The most

logical and common of these is the Gaussian

$$\phi = \exp((-\|u-c_i\|)^2/\sigma^2) \tag{5}$$

in which σ is a selectable scaling parameter.
In RBFN two sets of variables need to be settled on, the basis function centres, c_i, and the network weights w_i. The centres can be chosen in a number of ways. However, the centres, both in quantities and positions directly affect the efficiency and accuracy of the network. Essentially, it is best to place the centres where there is a considerable amount of activity in the input space. Unfortunately, if the input signals are random, or at least pseudo-random, this means that to adequately cover the input space in a reasonable way, a basis function centre is required for every data point. Obviously, for such a situation the RBFN approach becomes inviable where the input space is of high dimension, i.e. many randomly spaced signals, Narenda et al. [21]. However, as is the case in the operation of many systems, when the input signals are scattered in certain closely knit regions of the input space, in clear groupings or clusters, then the stronger are these clusters so the better they can be used of an RBFN. If the input vector is m-dimensional then so too is the cluster centre, m-dimensional.

In this report the RBFN is estimated using the `solverb` m-file in the Matlab Neural Networks Toolbox, Demuth et al. [19]. This algorithm employs an iterative design method, where neurons are created one at a time. At each iteration the input vector which will result in lowering the network error the most, is used to create a new neuron. The iteration continues until the error is below a user specified error goal or the number of neurons is equal to the maximum number of neurons. The latter cannot exceed the number of input vectors.

2.3 Recurrent Networks

The MLPN and RBFN described above belong to the static class of neural networks. Such networks are only capable of modelling dynamic systems if the inputs and outputs to the network are established from knowledge of the system to be identified. Therefore, in general, it is necessary to use artificial recurrent neural networks with feedback to model dynamic systems, see Hertz et al. [2], Sjöberg [12], Sørensen [22] and Kirkegård [23]. In the neural network terminology recurrent means that future network inputs will depend on present and past network outputs. This class of artificial neural networks with feedback, and therefore, inherently recursive is commonly referred to as partially or fully recurrent neural networks. A fully recurrent neural network has feedback between all the nodes. In the following an approach to incorporate feedback into a neural network in order to model a non-linear dynamic is described.

The recurrent neural network (RN) model is formulated by a Nonlinear Innovation State Space Model (ISSM) where p state variables are observed, see Sørensen [22].

$$\begin{aligned} \boldsymbol{X}(t) &= G(\boldsymbol{X}(t-\Delta t),\boldsymbol{U}(t-\Delta t),\boldsymbol{E}(t-\Delta t),\boldsymbol{W}) \\ \boldsymbol{Y}(t) &= \boldsymbol{H}\boldsymbol{X}(t) + \boldsymbol{E}(t) \end{aligned} \tag{6}$$

$X(t)$ and $U(t)$ are the estimate of the state vector and the excitation vector at the discrete time step t, respectively. $E(t)$ is the prediction error vector of order p at the discrete time step t. $Y(t)$ is the measured observation vector of order p at the discrete time step t. H is the observation matrix of order $p \times n$. Incomplete State Information may occur, i.e. $X(t)$ is not completely measurable. The matrix H can be chosen to $H = [I_{p,p}\ 0_{p,n-p}]$, where $I_{p,p}$ is a $p \times p$ unity matrix and $0_{p,n-p}$ is a $p \times n-p$ zero matrix. In this way the elements in $HX(t)$ are equal to the first p elements of the observation vector $Y(t)$.

In order to train an MLP to learn the Innovation State space model in (6) a network with the state vector, the excitation of the system and the prediction error vector are used as input and the state vector as output, see Sørensen [22]. This means that the desired outputs of the network are the elements of $Y(t)$. In this report the RN is estimated using the `nnssif` m-file in the Neural Network Based System Identification Toolbox, Nørgaard [24].

3. SURFACE SHIP MATHEMATICAL MODEL

3.1 The Son & Nomoto Model

The ship-model used is a 1st quadrant parametric model for a single-screw high-speed container ship in surge, sway, roll and yaw (4 parameter model) by Son and Nomoto, see Son & Nomoto [25], Fossen [26]. The models by Son and Nomoto are nonlinear equations of motion (surge, sway, roll, yaw), non-linear course-keeping equations of motion (sway, roll, yaw) and linearized course-keeping equations of motion (sway, roll, yaw). The equations used are listed in Fossen, [26], at pp.441 - 444. The implementation of the equations is listed in file `v07.m` (appendix A.0), from Lauvdal et al. [27].

3.2 Test Cases

The test cases are all made using the Matlab toolbox by T. Lauvdal & T.I. Fossen: "Matlab Simulation Program for Marine and Flight Vehicles", see Lauvdal et al. [27]. The high-speed container ship model (No. 07 in the above-mentioned toolbox) is used.

Various standard full-scale manoeuvring trials are defined and used to determine manoeuvring characteristics, like: turning circle, zig-zag, pull-out, spiral and reverse spiral manoeuvres and stopping trials. For this test four different test cases have been defined primarily in order to be able to test: a) how different NN models behave and b) how good they are at predicting and simulating (interpolated) new data.

Table 1 shows the test cases used for NN training. The test cases labelled x.0 have a duration of 1800 seconds (30 minutes) and test cases x.1 a duration 900 seconds (15 minutes), except test case 4.1 which has a duration of 1800 seconds. The x.0 test cases are used for training and test cases x.1 for testing.

TEST CASES		
Test Case	**File**	**Description**
1. "Right turn"		
1.0	TESTC_10.MAT	Rudder: -2° [60;120] , -6° [960;1020]
1.1	TESTC_11.MAT	Rudder: -4° [60;120]
2. "Turning circle"		
2.0	TESTC_20.MAT	Rudder: -6° [60;780] , -2° [1080;1800]
2.1	TESTC_21.MAT	Rudder: -4° [60;780]
3. "Right turn"		
3.0	TESTC_30.MAT	Rudder: -8° [180;240] , Speed: 15rpm [0,420] Rudder: -4° [780;840] , Speed: 15rpm [660,1020]
3.1	TESTC_31.MAT	Rudder: -6° [180;240] , Speed: 15rpm [0,420]
4. "Track sailing"		
4.0	TESTC_40.MAT	Rudder: -6° [60;120] , Rudder: 5° [300;480] Rudder: -3° [720;840] , Rudder: 4° [960;1080] Rudder: -1° [1080;1200] , Rudder: 2° [1200;1380] Rudder: 3° [1500;1680] , Speed : 50rpm [900;1800]
4.1	TESTC_41.MAT	Rudder: 3° [120;300] , Rudder: 2° [420;600] Rudder: -1° [600;720] , Rudder: 4° [720;840] Rudder: -3° [960;1080] , Rudder: 5° [1320;1500] Rudder: -6° [1680;1740] , Speed : 50rpm [0;900]

Table 1. Test cases used for NN training (Son & Nomoto ship model). All interval values are given in seconds. Outside the intervals listed the rudder angle is 0 and the shaft speed is 65 rpm.

All simulations are done using the Lauvdal & Fossen toolbox using an interval of 0.1 second in the Euler integration and are all verified by comparison with the results obtained using the file `ode34` in Matlab. The number of data in the test cases have been decreased so that only every 30th sample point (1 per three seconds) is used in the NN training. This is equivalent to 10 samples for each period in e.g. the 3rd or 5th element in the input vector (roll velocity and roll angle).

The detailed specifications of test cases are listed in Appendix A.0 (file `v07_cont.m`).

The initial state of the ship is for all test cases: the actual shaft speed is 65 rpm and the surge velocity is 6.806 m/s (which is equivalent to the steady-state speed when the shaft speed is 65 rpm).

All other initial values in the state vector are equal to zero. The state vector has 12 elements which are all listed below (taken from file `v07_data.m`). The intervals for rudder angle and shaft speed are likewise defined in the file.

File: `v07_data.m`

```
% V07_DATA   datafile for v07.m. Contains initial state
%            vector, initial input and main dimensions.
%
% NTH 1994 Trygve Lauvdal
clc
echo on

% Default parameters for High Speed Container Ship :
%
% Max rudder angle    : 10 deg
% Max rudder rate     : 5 deg/s
% Max shaft velocity  : 160 rpm
% Length of ship      : 175 m

% Initial states,inputs and sample time (to be edited by user):

u     = 6.8060;        % surge velocity         (m/s)
v     = 0;             % sway velocity          (m/s)
p     = 0;             % roll velocity          (rad/s)
r     = 0;             % yaw velocity           (rad/s)
phi   = 0;             % roll angle             (rad)
psi   = 0;             % yaw angle              (rad)
xpos  = 0;             % surge position         (m)
ypos  = 0;             % sway position          (m)
delta = 0;             % actual rudder angle    (rad)
n     = 65;            % actual shaft speed     (rpm)

d_c   = 10*pi/180;     % commanded rudder angle (rad)
n_c   = 0;             % commanded shaft speed  (rpm)

h     = 0.10;           % sample time            (s)

x     = [u  v  p  r  phi psi xpos ypos delta n]';   % state vector
u     = [d_c  n_c]';                                % input vector

echo

current = pwd;
if ~exist(['sim.m']),
saveon  = [current '/v07_data x u h'];
else
saveon  = [current '/v07_data x u h'];
end

eval(['save ' saveon])

%
% End
```

3.3 Input/Output Specification

The input/output vectors for all the networks in section 4 are specified in table 2 below.

No.	Input	Output
1	u surge velocity	u surge velocity
2	v sway velocity	v sway velocity
3	p roll velocity	p roll velocity
4	r yaw velocity	r yaw velocity
5	ϕ roll angle	ϕ roll angle
6	ψ yaw angle	ψ yaw angle
7	δ actual rudder angle	
8	n actual shaft speed	

Table 2. Input /output state vectors for all the networks in section 4.

4. Results

This chapter presents the results obtained using the three neural network approaches outlined in chapter 2. The charts showing the results using MLPN are all listed in Appendix A.1. The results using RBFN (section 4.2) and RN (section 4.3) are listed in appendices A.2 and A.3, respectively.

4.1 Multi-Layer Perceptron

An identical network topology is used for all four test-cases based on different trials trained with test case 4. A four layer network with 8 input neurons, 6 output neurons and two hidden layers with 25 and 20 neurons is chosen. A simple scaling of all I/O is applied so that the largest value in each of the 8 input vectors is equal to 1.0.

The "stopcriteria" shown in figures for training using backpropagation lists two numbers like "0.10 (0.22)" (see figure [m1a]). The first number is the requested stopping criteria and the second the value of the term used as stopping criteria reached after the number of epochs listed, i.e. each pass through all of the training input and output vectors. The stopping criteria is the summed square of the error between the real output and the output predicted by the network.

Better results may be obtained using a different scaling (e.g. each input vector normalized to a

specific length). Different scaling of the input was tested on test case 4 without any significant changes in convergence speed or in prediction or simulation results. Different network topologies having more neurons in the hidden layer were tried using test case 4 without significant better results.

4.1.1 Test Case 1

Figures [m1b] and [m1c] show that prediction for both known and unknown data works satisfactory. The prediction of the surge velocity is wrong and the oscillating trend from the roll velocity and roll angle is clearly recognisable in the surge velocity.

The simulation results have no similarities with the real data neither for the data used for training [m1b] or unknown data [m1d]. The bad simulation results are due to the bad modelling and narrow training interval for the surge velocity where small errors cause the surge velocity to leave the interval used for training and thereby requiring the NN model to extrapolate the surge velocity which is not working here. If the surge velocity is removed (i.e. predicted) the simulation shows fairly good results.

See also general observations for all four test cases in section 4.1.4.

4.1.2 Test Case 2

The same comments as for test case 1 are applicable for test case 2. The main reason for the wrong simulation results is again the bad modelling of the surge velocity, see e.g. figure [m2b] where the surge velocity goes to +12 m/s which is far from the training interval [6;7] m/s for the surge velocity.

4.1.3 Test Case 3

Same comments as for test case 1 and 2 are applicable here. Simulation results are only slightly better and only comparable for the first 0-60 seconds following that the error accumulates and the simulation results are meaningless. The prediction results using unknown data [m3c] are very good giving small errors for the whole 900 seconds.

4.1.4 Test Case 4

From figure [m4a] it is seen that the network is able to model the state vector satisfactorily except for the surge velocity which shows large errors both in magnitude and form. It is also visible from the error charts directly that there is a correlation between the signal and the errors (compare e.g. "peaks" on "yaw velocity" and the corresponding error plot). Also, the autocorrelation function for the error is far from being a white noise approximation.

From figure [m4b] it is seen that simulation only gives usable results during the first 90 seconds whereafter the results are not usable at all. The reason for the errors using simulation is primarily the bad modelling of the surge velocity (see figure [4ba]). If the surge velocity in the simulation is input using the real values the simulation gives fairly good results, see below.

Figure [m4c] shows that prediction using new data is good (except for the last 500 seconds where the real and predicted curves diverges). [m4d] shows that simulation with new data is as bad as simulation with the trained data. Again, simulation without the surge velocity gives usable results.

The simulation comparisons [m4b] and [m4d] are also shown in figures [m4b_] and [m4d_]. In the last two plots the surge velocity is predicted to avoid the error from the surge velocity. As it can be seen the simulation results are a lot better here than in [m4b] and [m4d].

This network topology has been tested by running the backpropagation training until convergence in the stopping criteria. It was found that additional epochs above 30000 for test case 4 gave no decrease in the stopping criteria term.

4.2 Radial Basis Networks

The RBFN has been designed using a three layer network with 8 input neurons, 6 linear output neurons a one hidden layer with Gaussian neurons. The "stopcriteria" shown in figures for training using RBFN lists two numbers like "0.20 (0.20)" , see figure [r1a]. The first number is the requested stopping criteria and the second the value of the term used as stopping criteria. The RBFN implementation, see Demuth et al. [19] employs an iterative design method, where neurons are created one at a time. At each iteration the input vector which will result in lowering the network error the most, is used to create a new neuron. The iteration continues until the error is below the stopcriteria or the number of neurons is equal to the maximum number of neurons. This means that number of epochs refers to number of neurons for `solvebr`. The different user specified parameters have been selected by a trial and error study.

4.2.1 Test Case 1

The results in figures [r1a] and [r1c] show that the RBFN gives very well predictions for both known as well as unknown data. However, the prediction errors indicate that the obtained RBFN does not model the data satisfactorily due to the missing white noise prediction error sequences.

The simulation results [r1b] and [r1d] obtained by the RBFN can be given the same comments as the simulation results obtained using MLPN in test case 4.1.1

4.2.2 Test Case 2

Here, the same comments as given in section 4.1.1, 4.1.2, 4.1.3 and 4.2.1 can be given.

4.2.3 Test Case 3

Again, the same comments as given in section 4.2.2 can be used. However, it seems as the simulation results are slightly better but not usable. Further, it is seen that the RBFN uses 58 neurons/nodes in the hidden layer while only 6 and 18, respectively, have been used in the test cases 4.2.1 and 4.2.2.

4.2.4 Test Case 4

The results presented in figures [r4a] and [r4c] show that the RBFN models the training data very well but the prediction results using data not included in the training data are not satisfactory. The simulation results in this test case are as bad as for the other test cases with RBFN.

4.3 Recurrent Networks

Two different RN topologies are selected for the four test-cases based on different trials training. A three layered RN with 14 input neurons, 6 output neurons and a hidden layer with 6 neurons is chosen for the test cases 1 and 2 while 12 neurons are used in the hidden layer for the test cases 3 and 4, respectively. Before training the data have been scaled.

Number of iteration used for training the RN is listed together with the "stopcriteria" information, and topology information at the charts shown in appendix 3.

4.3.1 Test Case 1

Figures [i1a] and [i1c] show that prediction for both known and unknown data works satisfactory. However, as observed for the MLPN and the RBFN, respectively the simulation results are not useable.

4.3.2 Test Case 2

The results from this case again show that usable prediction results can be obtained but the simulation results are not usable.

4.3.3 Test Case 3

This test case again shows satisfactorily prediction results and unsatisfactory simulation results. Further, it is seen that 12 neurons are used in this test case.

4.3.4 Test Case 4

From figures [i4a] and [i4c], respectively it is seen that the network is able to model the state vector satisfactorily. However, as for all the other test cases the error is far from being a white noise approximation.

From figures [i4b] and [i4d] it is seen that simulation only gives results which are slightly usable for some of the state variables, see e.g. sway and yaw. Especially, the results in figure [i4d] seem to be interesting. One should expect that these simulations could be quite good if the number of training data was increased in order to model the variation in e.g. surge better. However, these investigations will not be made in this report.

4.4 Summary

Firstly, it must be emphasized that no direct comparison (in order to determine the "best" network topology and training method) between the feasibility and usability of the three different neural networks and their behaviour can be made. The three different network topologies and different methods of training, e.g. network topology, network size and amount of training are not directly comparable.

Comparing prediction results using known data, e.g. [m1a], [r1a] and [i1a] do not show any significant difference. MLPN seems slightly better than RN and RBFN. The difference may be due to different network topologies and training. More interesting is the prediction behaviour using unknown data. Comparing e.g. [m1c], [r1c] and [i1c] show better results using MLPN than RN and RBFN. For RBFN [i1c] the prediction results are influenced by a high-frequency error which makes prediction of roll unusable. However, this high frequency is less dominating in [i2c], [i3c] and [i4c]. Simulation results, see e.g. [m1b], [r1b] and [i1b] are not usable at all, neither used on trained data or unknown data.

In general it seems that prediction with unknown data gave best results for MLPN and RBFN. The RN has difficulties modelling the non-oscillating behaviour of sway and yaw (see e.g. [i2c]) whereas MLPN and RBFN show no oscillations in prediction of sway and yaw (see e.g. [m2c] and [r2c]). Looking at figures [m3c], [r3c] and [i3c] it is seen that the RBFN has problems modelling the correct behaviour at t = 420 seconds where the shaft velocity changes - this problem does not occur for MLPN and RN.

Prediction test using unknown data for test case 4 ([m4c], [r4c] and [i4c]) shows that MLPN have small difficulties (surge velocity) modelling the last 300 seconds, the RBFN again have difficulties when the shaft velocity changes and RN has some high-oscillation errors also encountered in [i1c].

Overall, prediction using unknown data is generally good. Some irregularities occur using RBFN (sudden large prediction errors) and using RN (high-frequency oscillations). Simulation using any of the three training networks is in general not usable. However, simulation where one or more of the elements in the input vector is predicted (surge velocity) gives usable results (see e.g. [m4b_] and [m4d_]. Also, starting simulation at an arbitrary point in time will give good simulation results for 0-60 seconds before the simulation results diverge and become unusable.

Looking back at the test cases a number of unfavourable issues may be pointed out. The test cases used for training are too limited and should have been expanded to cover a broader interval of the elements in the input and output vectors. Also, tests using simulation should not start when the ship is in a "steady state" (roll angle and yaw angle equal to zero).

The results show, however, that prediction is usable and that simulation results may be used to get a correct modelling of the behaviour of the ship for a limited time period after start of a simulation or a prediction.

5. CONCLUSIONS

This report contains results of an investigation using different types of neural networks to predict the surge, sway, roll and yaw behaviour of a single-screw, high-speed container ship.

The experience obtained and errors done in this report will be taken into account in the further work in the project "Application of Neural Networks in Numerical Modelling of a Human Ship Navigator".

6. ACKNOWLEDGEMENTS

The present research was supported by The Danish Technical Research Council within the project: "Marine Structural Design ".

7. REFERENCES

[1] McCulloch, W.S. W. Pitts: A Logical Calculus of the Ideas Immanent in Nervous Activity. Bull. Math. Biophys. Vol. 5, pp. 115-133, 1943.

[2] Hush, D.R. B.G. Horne: Progress in Supervised Neural Networks. IEEE Signal Processing Magazine, January, 1993.

[3] Hertz, J., A. Krogh R.G. Palmer: Introduction to the Theory of Neural Computation. Addison-Wesley, Redwood City, CA, 1991.

[4] Hopfield, J.J: Neural Networks and Physical Systems with Emergent Collective Computational Abilities. Proc. Nat. Acad. Sci. Vol. 79, pp. 2554-2558, 1982.

[5] Kohonen, T.: Self-Organization and Associative Memory. Springer-Verlag,1984.

[6] Rumelhart, D.E. J.L. McClelland: Parallel Distributed Processing: Explorations in the Microstructure of Cognition, Vol. 1 Foundations. Cambridge, MIT Press,1986.

[7] Masri, S.F., A.G. Chassiakos T.K. Caughey: Structure-unknown Non-Linear Dynamic Systems: Identification Through Neural Networks. Smart Mater.Struct. Vol. 1,pp. 45-56, 1992.

[8] Qi, G.Z., H.M. Chen, K.H. Tsai J.C.S. Yang: Structural Dynamic Model Identification Through Neural Network. Proceedings of the 12th Int. Modal AnalysisConf., pp. 1167-1172, 1994.

[9] Riva, A., L. Garibaldi, E. Giorcelli A. Fasana: Dynamic System Identification by means of Neural Network. Proceedings of the 11th Int. Modal Analysis Conf., pp. 928-933, 1993.

[10] Seidl, R.D. R.D. Lorenz: A Structure by which a Recurrent Neural Network Can

Approximate a Nonlinear Dynamic System. IJCNN, Seattle, 1991.

[11] Pham,D.T. S.J. Oh: A Recurrent Backpropagation Neural Network for Dynamic System Identification. Journal of Systems Engineering, Vol. 2, 1992.

[12] Sjöberg,J. : Non-linear Identification with Neural Network. Ph.D Thesis, Linköbing University, Department of Electrical Engineering, May, 1995.

[13] Burns, R.S.: The Use of Artificial Neural Networks for the Intelligent Optimal Control of Surface Ships. IEEE Journal of Oceanic Engineering, Vol. 20, No.1, January 1995.

[14] Endo, M, J. van Amerongen & A.W.P. Bakkers: Applicability of Neural Networks to Ship Steering. Proc. of the 3rd IFAC Workshop on Control Applications in Marine Systems (ed. T.I. Fossen), Trondheim, 1995

[15] Witt, N.A.J, R. Sutton & K.M. Miller: A Track Keeping Neural Network Controller for Ship Guidance. Proc. of the 3rd IFAC Workshop on Control Applications in Marine Systems (ed. T.I. Fossen), Trondheim, 1995

[16] Balasuriya, B.A.A.P. & P.R.R Hoole: Feedforward Neural-Network Controller for Ship Steering. Proc. of the 3rd IFAC Workshop on Control Applications in Marine Systems (ed. T.I. Fossen), Trondheim, 1995

[17] Topping, B.H.V: Proc. Of the Sixth International Conference on Civil and Structural Engineering. Edinburgh, 1993.

[18] Billings, S.A. & H.B. Jamaluddin: A Comparison of the Backpropagation and Recursive Prediction Error Algorithms for Training Neural Networks. Mechanical Systems and Signal Processing, Vol. 5, pp. 233-255, 1991.

[19] Demuth, H & M. Beale: Neural Network Toolbox, V.20. The Mathworks, Inc. 1994.

[20] Chen, S., C.F.N. Cowan & P.M. Grant: Orthogonal Least Squares Learning Algorithm for Radial Basis Function Networks. IEEE Transactions on Neural Networks, Vol. 2, No. 2, pp. 302-309, March 1991.

[21] Narendra, K.S.: Adaptive Control of Dynamical Systems using Neural Networks. Van Nostrand, 1994

[22] Sorensen, O.: Neural Networks in Control Applications. Ph.D-thesis from Aalborg University, 1994.

[23] Kirkegaard,P.H., S.R.K. Nielsen & H.I. Hansen: Identification of Non-Linear Structures using Recurrent Neural Networks. Proc. of the 13th International Modal Analysis Conference, Nashville, 1995

[24] Nørgaard, M. Neural Network Based System Identification Toolbox. V1.0, Institute of

Automation, Electronics Institute, Institute of Mathematical Modelling, Technical University of Denmark, 1995.

[25] Son & Nomoto: "On the Coupled Motion of Steering and Rolling of a High Speed Container Ship", Naval Architect of Ocean Engineering, 20:73-83. From J.S.N.A. , Japan, Vol. 150, 1981.

[26] Fossen, T.I: Guidance and Control of Ocean Vehicles. Wiley, New York, 1994.

[27] Lauvdal, T. & T.I. Fossen: Documentaion for Matlab Simulation Program for Marine and Flight Vehicles. NTH report 94-43-W, University of Trondheim, Department of Engineering Cybernetics, March, 1995.

Appendix A. 0

This appendix contains the files used for generation of test cases (`v07_cont.m`) and the file containing the implementation of the Son & Nomoto model (`v07.m`). Both from Fossen [26],

File: `v07_cont.m`

```
%-------------------------------------------------------------------------
function u1 = V07_cont(k,x,u)
u1 = [ 0*pi/180; 65 ];
%
icase=10;
% ------------------------------------------------------------
% Test case 10: Right turn: 2 deg. for 1 minute [ 1;  2]
%               Right turn: 6 deg. for 1 minute [16; 17]
% 30 minutes, delta_t = 3 sec., 600 samples
%
if (icase==10)
  if (k>60)
    u1 = [ -02*pi/180;  65];
  end;
  if (k>120)
    u1 = [ 0*pi/180;  65];
  end;
  if (k>960)
    u1 = [ -06*pi/180;  65];
  end;
  if (k>1020)
    u1 = [ 0*pi/180;  65];
  end;
end;
% ------------------------------------------------------------
% Test case 11: Right turn: 4 deg. for 1 minute [1 ; 2]
% 15 minutes, delta_t = 3 sec., 300 samples
%
if (icase==11)
  if (k>60)
    u1 = [ -04*pi/180;  65];
  end;
  if (k>120)
    u1 = [ 0*pi/180;  65];
  end;
end;
% ------------------------------------------------------------
% Test case 20: Turning circle: 6 deg. for 12 minutes [ 1 ; 13]
%               Turning circle: 2 deg  for 12 minutes [18 : 30]
% 30 minutes, delta_t = 3 sec., 600 samples
%
if (icase==20)
  if (k>60)
    u1 = [ -06*pi/180;  65];
  end;
  if (k>780)
    u1 = [ -00*pi/180;  65];
  end;
  if (k>1080)
    u1 = [ -02*pi/180;  65];
  end;
  if (k>1800)
    u1 = [ -00*pi/180;  65];
  end;
```

```
end;
% ------------------------------------------------------------
% Test case 21: Turning circle: 4 deg. for 12 minutes [1 ; 13]
% 20 minutes, delta_t = 3 sec., 400 samples
%
if (icase==21)
  if (k>60)
    u1 = [ -04*pi/180;  65];
  end;
  if (k>780)
    u1 = [ -00*pi/180;  65];
  end;
end;
% ------------------------------------------------------------
% Test case 30:  Slow speed + Right turn + normal speed
% 30 minutes, delta_t = 3 sec., 600 samples
%
if (icase==30)
  if (k>0)
    u1 = [ -00*pi/180;  15];
  end;
  if (k>180)
    u1 = [ -08*pi/180;  15];
  end;
  if (k>240)
    u1 = [ -00*pi/180;  15];
  end;
  if (k>420)
    u1 = [ -00*pi/180;  65];
  end;
  if (k>660)
    u1 = [ -00*pi/180;  15];
  end;
  if (k>780)
    u1 = [ -04*pi/180;  15];
  end;
  if (k>840)
    u1 = [ -00*pi/180;  15];
  end;
  if (k>1020)
    u1 = [ -00*pi/180;  65];
  end;
end;
% ------------------------------------------------------------
% Test case 31:  Slow speed + Right turn + normal speed
% 15 minutes, delta_t = 1 sec., 300 samples
%
if (icase==31)
  if (k>0)
    u1 = [ -00*pi/180;  15];
  end;
  if (k>180)
    u1 = [ -06*pi/180;  15];
  end;
  if (k>240)
    u1 = [ -00*pi/180;  15];
  end;
  if (k>420)
    u1 = [ -00*pi/180;  65];
  end;
end;
% ------------------------------------------------------------
% Test case 40:  "track"
```

```
% 20 minutes, delta_t = 1 sec., 1200 samples
%
if (icase==40)
  if (k>0)
    u1 = [ -00*pi/180;  65];
  end;
  if (k>60)
    u1 = [ -06*pi/180;  65];
  end;
  if (k>120)
    u1 = [ -00*pi/180;  65];
  end;
  if (k>300)
    u1 = [ +05*pi/180;  65];
  end;
  if (k>480)
    u1 = [ +00*pi/180;  65];
  end;
  if (k>720)
    u1 = [ -03*pi/180;  65];
  end;
  if (k>840)
    u1 = [ -00*pi/180;  65];
  end;
  if (k>900)
    u1 = [ -00*pi/180;  50];
  end;
  if (k>960)
    u1 = [  04*pi/180;  50];
  end;
  if (k>1080)
    u1 = [ -01*pi/180;  50];
  end;
  if (k>1200)
    u1 = [  02*pi/180;  50];
  end;
  if (k>1380)
    u1 = [  00*pi/180;  50];
  end;
  if (k>1500)
    u1 = [  03*pi/180;  50];
  end;
  if (k>1680)
    u1 = [  00*pi/180;  50];
  end;
end;
% ------------------------------------------------------------
% Test case 41:  "track" / reverse !
% 30 minutes, delta_t = 3 sec., 600 samples
%
if (icase==41)
  if (k>0)
    u1 = [ -00*pi/180;  50];
  end;
  if (k>120)
    u1 = [  03*pi/180;  50];
  end;
  if (k>300)
    u1 = [ -00*pi/180;  50];
  end;
  if (k>420)
    u1 = [ +02*pi/180;  50];
  end;
```

```
    if (k>600)
      u1 = [ -01*pi/180;  50];
    end;
    if (k>720)
      u1 = [  04*pi/180;  50];
    end;
    if (k>840)
      u1 = [ -00*pi/180;  50];
    end;
    if (k>900)
      u1 = [ -00*pi/180;  65];
    end;
    if (k>960)
      u1 = [ -03*pi/180;  65];
    end;
    if (k>1080)
      u1 = [ -00*pi/180;  65];
    end;
    if (k>1320)
      u1 = [  05*pi/180;  65];
    end;
    if (k>1500)
      u1 = [  00*pi/180;  65];
    end;
    if (k>1680)
      u1 = [ -06*pi/180;  65];
    end;
    if (k>1740)
      u1 = [  00*pi/180;  65];
    end;
end;
%
% End
%
```

File: v07.m

```
function xdot = V07(x,ui)
%
% xdot = V07(x,ui); returns the time derivated of the state vector :
%
% x   = [ u v p r phi psi xpos ypos delta n ]'  where
%
% u     = surge velocity          (m/s)
% v     = sway velocity           (m/s)
% p     = roll velocity           (rad/s)
% r     = yaw velocity            (rad/s)
% phi   = roll angle              (rad)
% psi   = yaw angle               (rad)
% xpos  = position in x-direction (m)
% ypos  = position in y-direction (m)
% delta = actual rudder angle     (rad)
% n     = actual shaft velocity   (rpm)
%
% The input vector is :
%
% ui      = [ delta_c n_c ]'  where
%
% delta_c = commanded rudder angle   (rad)
% n_c     = commanded shaft velocity (rpm)
%
% NTH 1994, Trygve Lauvdal
%
```

```
%
% Reference : Son & Nomoto (1982).
%             On the Coupled Motion of Steering and
%             Rolling of a High Speed Container Ship,
%             Naval Architect of Ocean Engineering,
%             20: 73-83. From J.S.N.A. , Japan, Vol. 150, 1981.

% Check of input and state dimensions

if ~(length(x) == 10),error('x-vector must have dimension 10 !');end
if ~(length(ui) == 2),error('u-vector must have dimension  2 !');end

% Normalization variables

L = 175;                        % length of ship (m)
sqrV = (x(1)^2 + x(2)^2);
V    = sqrt(sqrV);              % service speed (m/s)

% Check of service speed

if V == 0,error('The ship must have speed greater than zero');end

delta_max  = 10;                % max rudder angle (deg)
Ddelta_max = 5;                 % max rudder rate (deg/s)
n_max      = 160;               % max shaft velocity (rpm)

% Non-dimensional states and inputs

delta_c = ui(1);
n_c     = ui(2)/60*L/V;

u     = x(1)/V;    v   = x(2)/V;
p     = x(3)*L/V;  r   = x(4)*L/V;
phi   = x(5);      psi = x(6);
delta = x(9);      n   = x(10)/60*L/V;

% Parameters, hydrodynamic derivatives and main dimensions

m  = 0.00792;     mx     = 0.000238;   my = 0.000238;
Ix = 0.0000176;   alphay = 0.05;       lx = 0.0313;
ly = 0.0313;      Ix     = 0.0000176;  Iz = 0.000456;
Jx = 0.0000034;   Jz     = 0.000419;   xG = 0;

B     = 25.40;    dF = 8.00;    g     = 9.81;
dA    = 9.00;     d  = 8.50;    nabla = 21222;
KM    = 10.39;    KB = 4.6154;  AR    = 33.0376;
Delta = 1.8219;   D  = 6.533;   GM    = .3/L;
rho   = 1000;     t  = 0.175;   T     = 0.0005;

W     = rho*g*nabla/(rho*L^2*V^2/2);

Xuu       = -0.0004226;  Xvr     = -0.00311;    Xrr       = 0.00020;
Xphiphi   = -0.00020;    Xvv     = -0.00386;

Kv        =  0.0003026;  Kr      = -0.000063;   Kp        = -0.0000075;
Kphi      = -0.000021;   Kvvv    =  0.002843;   Krrr      = -0.0000462;
Kvvr      = -0.000588;   Kvrr    =  0.0010565;  Kvvphi    = -0.0012012;
Kvphiphi  = -0.0000793;  Krrphi  = -0.000243;   Krphiphi  =  0.00003569;

Yv        = -0.0116;     Yr      =  0.00242;    Yp        =  0;
Yphi      = -0.000063;   Yvvv    = -0.109;      Yrrr      =  0.00177;
Yvvr      =  0.0214;     Yvrr    = -0.0405;     Yvvphi    =  0.04605;
Yvphiphi  =  0.00304;    Yrrphi  =  0.009325;   Yrphiphi  = -0.001368;
```

```
Nv       = -0.0038545;  Nr     = -0.00222;    Np       =  0.000213;
Nphi     = -0.0001424;  Nvvv   =  0.001492;   Nrrr     = -0.00229;
Nvvr     = -0.0424;     Nvrr   =  0.00156;    Nvvphi   = -0.019058;
Nvphiphi = -0.0053766;  Nrrphi = -0.0038592;  Nrphiphi =  0.0024195;

kk     =  0.631;  epsilon =  0.921;  xR    = -0.5;
wp     =  0.184;  tau     =  1.09;   xp    = -0.526;
cpv    =  0.0;    cpr     =  0.0;    ga    =  0.088;
cRr    = -0.156;  cRrrr   = -0.275;  cRrrv =  1.96;
cRX    =  0.71;   aH      =  0.237;  zR    =  0.033;
xH     = -0.48;

% Masses and moments of inertia

m11 = (m+mx);
m22 = (m+my);
m32 = -my*ly;
m42 = my*alphay;
m33 = (Ix+Jx);
m44 = (Iz+Jz);

% Rudder saturation and dynamics

if abs(delta_c) >= delta_max*pi/180,
   delta_c = sign(delta_c)*delta_max*pi/180;
end

delta_dot = delta_c - delta;
if abs(delta_dot) >= Ddelta_max*pi/180,
   delta_dot = sign(delta_dot)*Ddelta_max*pi/180;
end

% Shaft velocity saturation and dynamics

n_c = n_c*V/L;
n   = n*V/L;
if abs(n_c) >= n_max/60,
   n_c = sign(n_c)*n_max/60;
end

if n > 0.3,Tm=5.65/n;else,Tm=18.83;end
n_dot = 1/Tm*(n_c-n)*60;

% Calculation of state derivatives

  vR     = ga*v + cRr*r + cRrrr*r^3 + cRrrv*r^2*v;
  uP     = cos(v)*((1 - wp) + tau*((v + xp*r)^2 + cpv*v + cpr*r));
   J     = uP*V/(n*D);
  KT     = 0.527 - 0.455*J;
  uR     = uP*epsilon*sqrt(1 + 8*kk*KT/(pi*J^2));
  alphaR = delta + atan(vR/uR);
  FN     = - ((6.13*Delta)/(Delta + 2.25))*(AR/L^2)*(uR^2 + vR^2)*sin(alphaR);
  T      = 2*rho*D^4/(V^2*L^2*rho)*KT*n*abs(n);

% Forces and moments

  X    = Xuu*u^2 + (1-t)*T + Xvr*v*r + Xvv*v^2 + Xrr*r^2 + Xphiphi*phi^2 + ...
         cRX*FN*sin(delta) + (m + my)*v*r;

  Y    = Yv*v + Yr*r + Yp*p + Yphi*phi + Yvvv*v^3 + Yrrr*r^3 + Yvvr*v^2*r +
...
         Yvrr*v*r^2 + Yvvphi*v^2*phi + Yvphiphi*v*phi^2 + Yrrphi*r^2*phi + ...
         Yrphiphi*r*phi^2 + (1 + aH)*FN*cos(delta) - (m + mx)*u*r;
```

```
  K     = Kv*v + Kr*r + Kp*p + Kphi*phi + Kvvv*v^3 + Krrr*r^3 + Kvvr*v^2*r +
...
         Kvrr*v*r^2 + Kvvphi*v^2*phi + Kvphiphi*v*phi^2 + Krrphi*r^2*phi + ...
         Krphiphi*r*phi^2 - (1 + aH)*zR*FN*cos(delta) + mx*lx*u*r - W*GM*phi;


  N     = Nv*v + Nr*r + Np*p + Nphi*phi + Nvvv*v^3 + Nrrr*r^3 + Nvvr*v^2*r +
...
         Nvrr*v*r^2 + Nvvphi*v^2*phi + Nvphiphi*v*phi^2 + Nrrphi*r^2*phi + ...
         Nrphiphi*r*phi^2 + (xR + aH*xH)*FN*cos(delta);

% Dimensional state derivatives

xdot =[                          X*(V^2/L)/m11

-(-m33*m44*Y+m32*m44*K+m42*m33*N)/(m22*m33*m44-m32^2*m44-m42^2*m33)*(V^2/L)
(-m32*m44*Y+K*m22*m44-K*m42^2+m32*m42*N)/(m22*m33*m44-m32^2*m44-m42^2*m33)*
(V^2/L^2)
(-m42*m33*Y+m32*m42*K+N*m22*m33-N*m32^2)/(m22*m33*m44-m32^2*m44-m42^2*m33)*
(V^2/L^2)
                                    p*(V/L)
                                cos(phi)*r*(V/L)
                        (cos(psi)*u-sin(psi)*cos(phi)*v)*V
                        (sin(psi)*u+cos(psi)*cos(phi)*v)*V
                                  delta_dot
                                    n_dot                  ];
```

Appendix A.1 - MLPN Results

This Appendices A.1, A.2 and A.3, respectively present the obtained results from the examples in chapter 4.

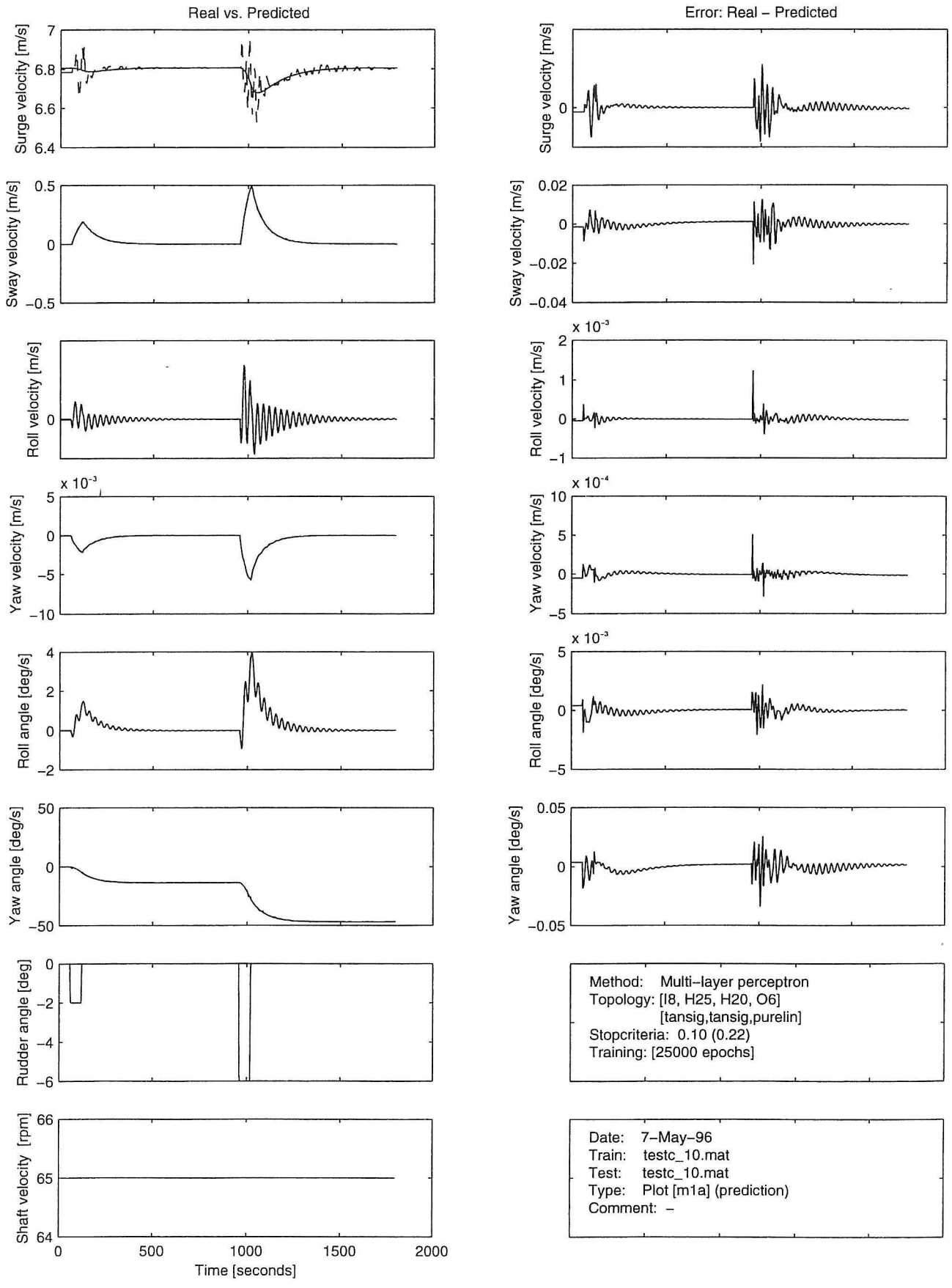
Four comparison charts are listed for each of the test cases 1 to 4 and for each of the three types of neural networks/neural network training.

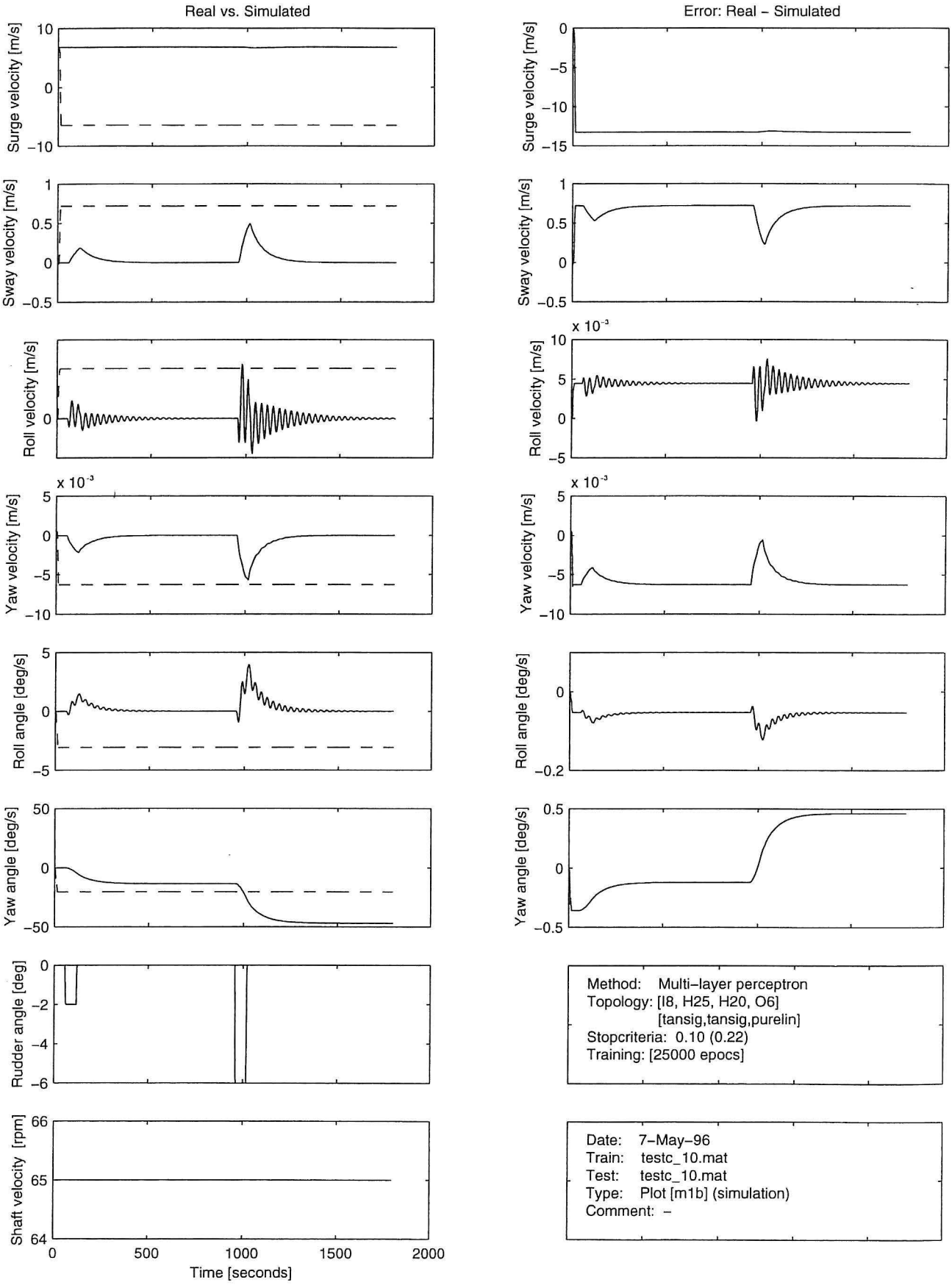
The charts here and in appendices A.2 and A.3 are labelled ([mri][1234][abcd]) where [mri] denotes MLPN, RBFN and RN, respectively, where [1234] denotes test case 1,2,3 or 4. The letters [abcd] denote a) prediction with trained data, b) simulation with trained data, c) prediction with unknown data and d) simulation with new data (see also table below).

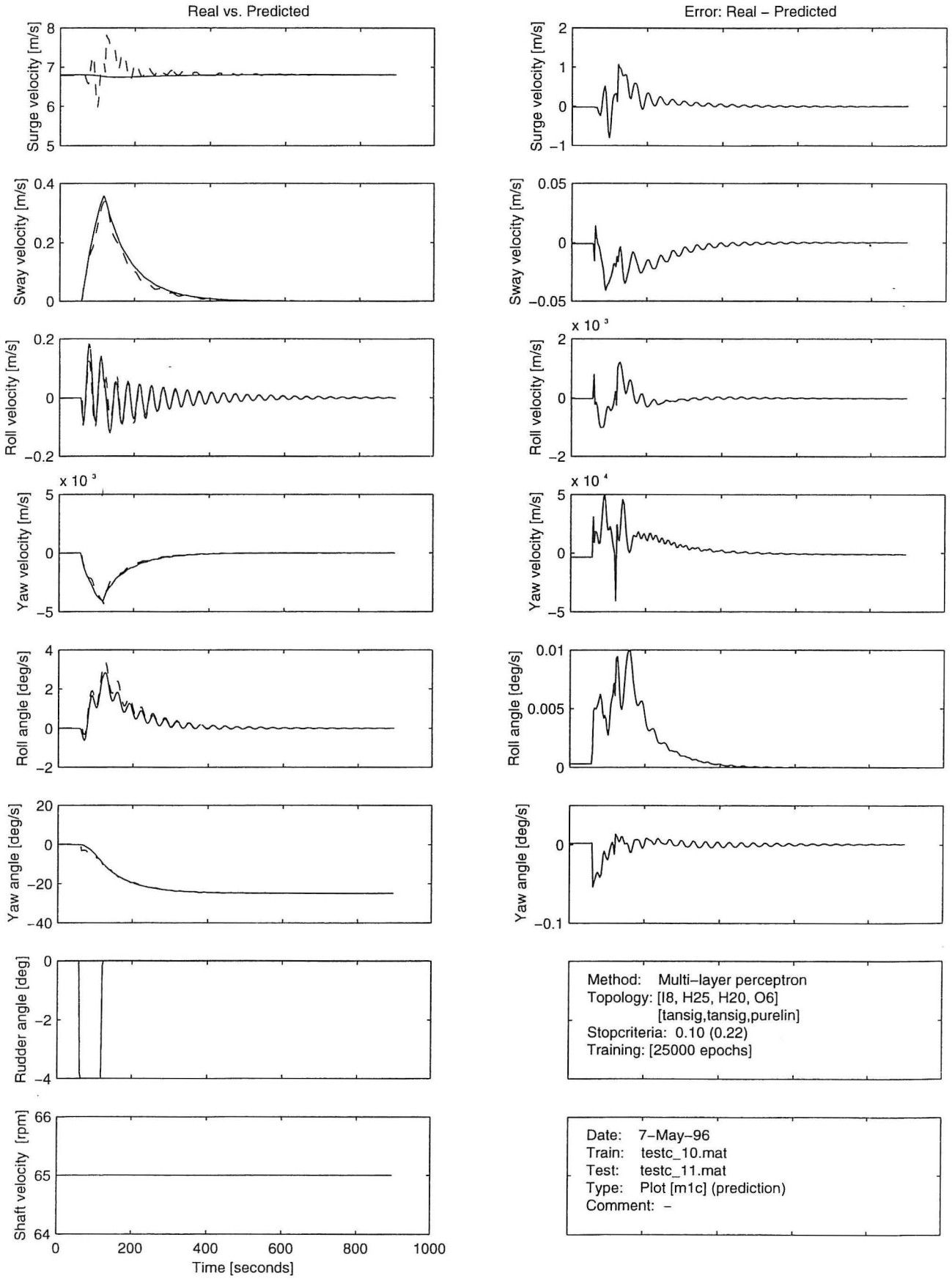
[abcd]	Trained data	Tested data	Prediction/Simulation
a	TESTC_x0.MAT	TESTC_x0.MAT	Prediction
b	TESTC_x0.MAT	TESTC_x0.MAT	Simulation
c	TESTC_x0.MAT	TESTC_x1.MAT	Prediction
d	TESTC_x0.MAT	TESTC_x1.MAT	Simulation

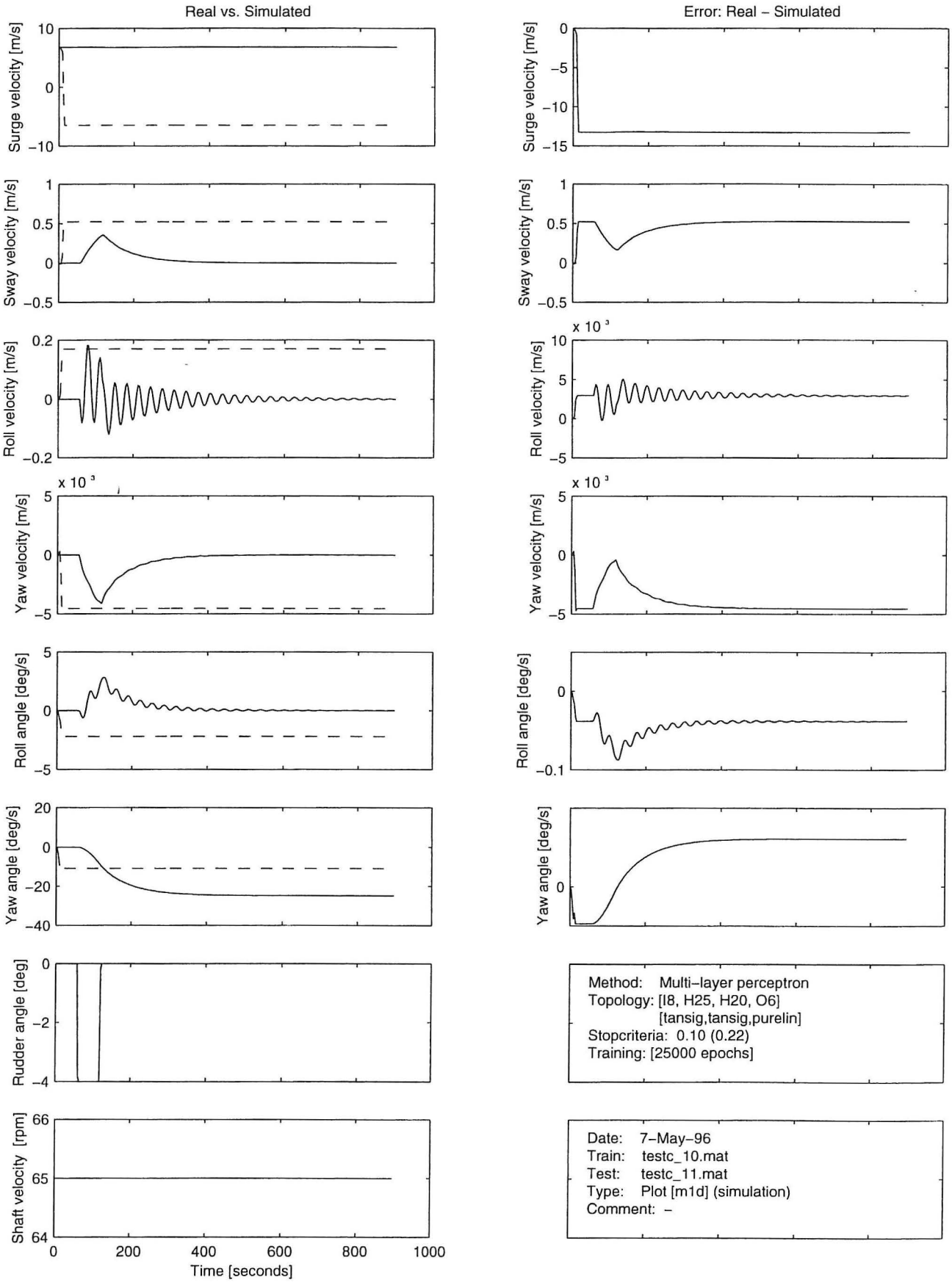
Table A.1 Numbering of charts with output from trained networks.

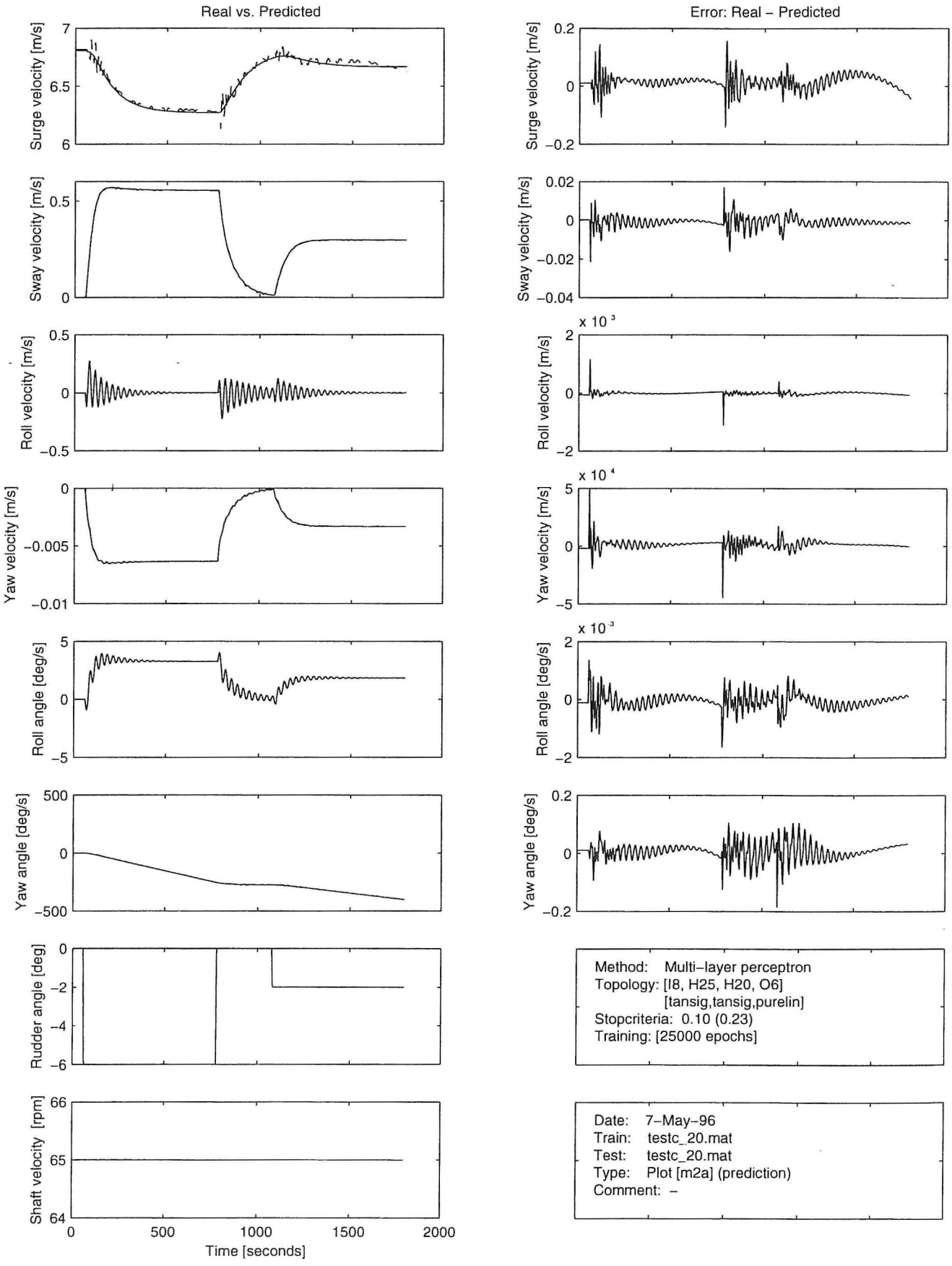
E.g. [m4a] is a neural network trained using MLPN "m" used on test case four "4" showing the prediction error using TESTC_40.MAT with a neural network trained using TESTC_40.MAT ("a",see table A.1.)

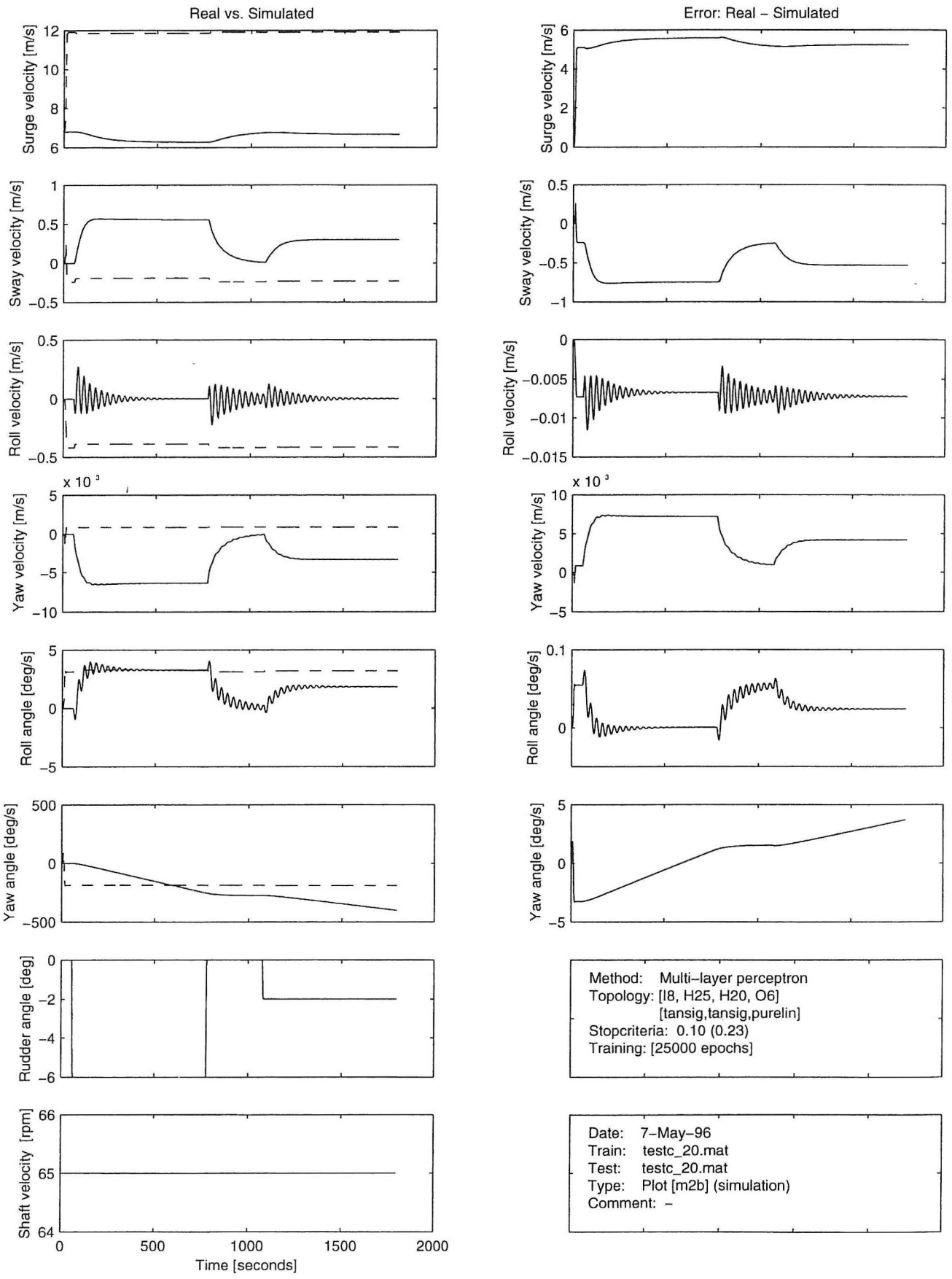

Real vs. Predicted
Error: Real – Predicted
Surge velocity [m/s]
Sway velocity [m/s]
Roll velocity [m/s]
Yaw velocity [m/s]
Roll angle [deg/s]
Yaw angle [deg/s]
Rudder angle [deg]
Shaft velocity [rpm]
Time [seconds]
Method: Multi-layer perceptron
Topology: [I8, H25, H20, O6]
[tansig,tansig,purelin]
Stopcriteria: 0.10 (0.22)
Training: [25000 epochs]
Date: 7-May-96
Train: testc_10.mat
Test: testc_10.mat
Type: Plot [m1a] (prediction)
Comment: –

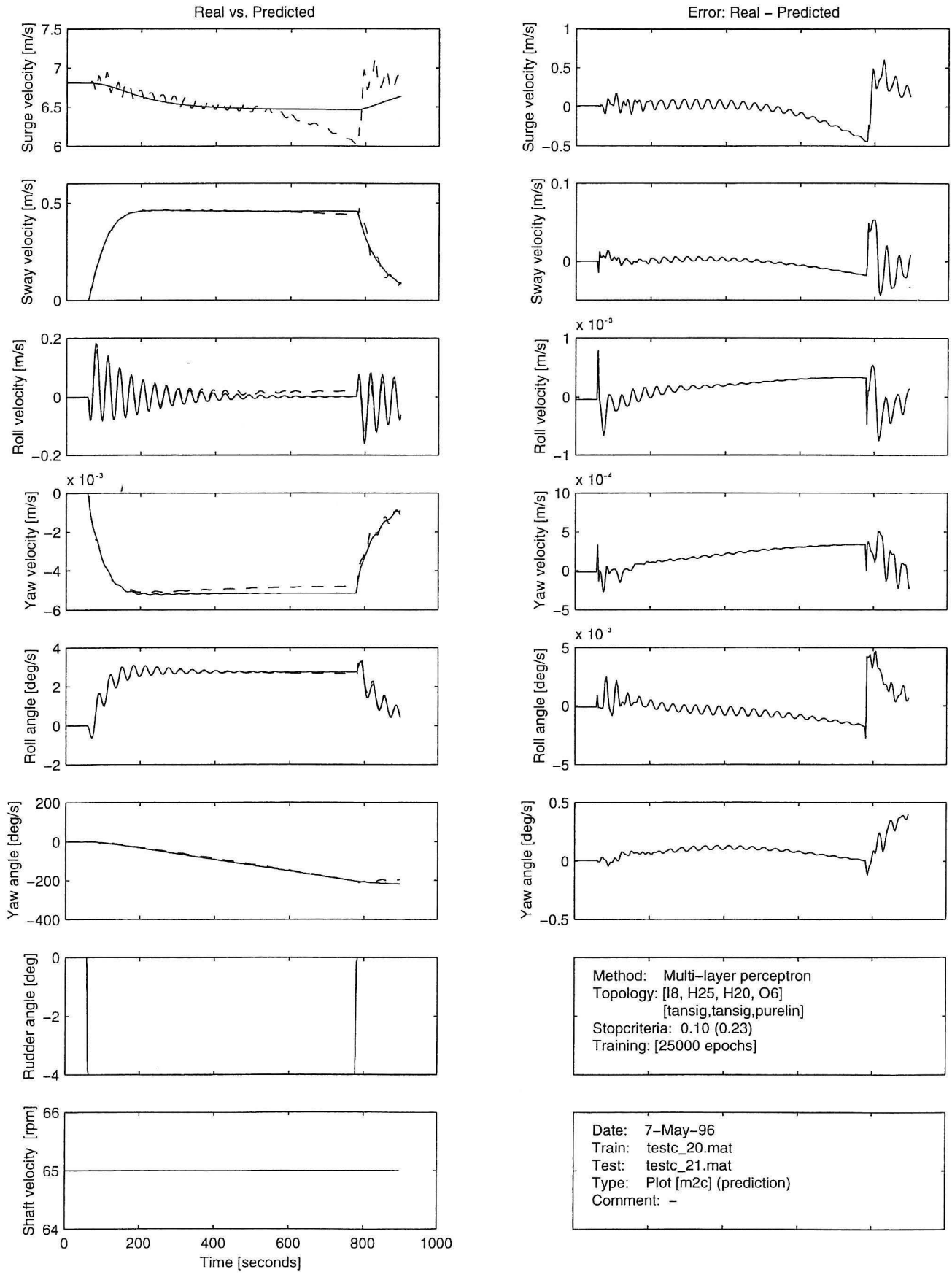

Real vs. Simulated
Error: Real – Simulated
Surge velocity [m/s]
Sway velocity [m/s]
Roll velocity [m/s]
Yaw velocity [m/s]
Roll angle [deg/s]
Yaw angle [deg/s]
Rudder angle [deg]
Shaft velocity [rpm]
x 10⁻³
Time [seconds]
Method: Multi-layer perceptron
Topology: [I8, H25, H20, O6]
[tansig,tansig,purelin]
Stopcriteria: 0.10 (0.22)
Training: [25000 epocs]
Date: 7-May-96
Train: testc_10.mat
Test: testc_10.mat
Type: Plot [m1b] (simulation)
Comment: –

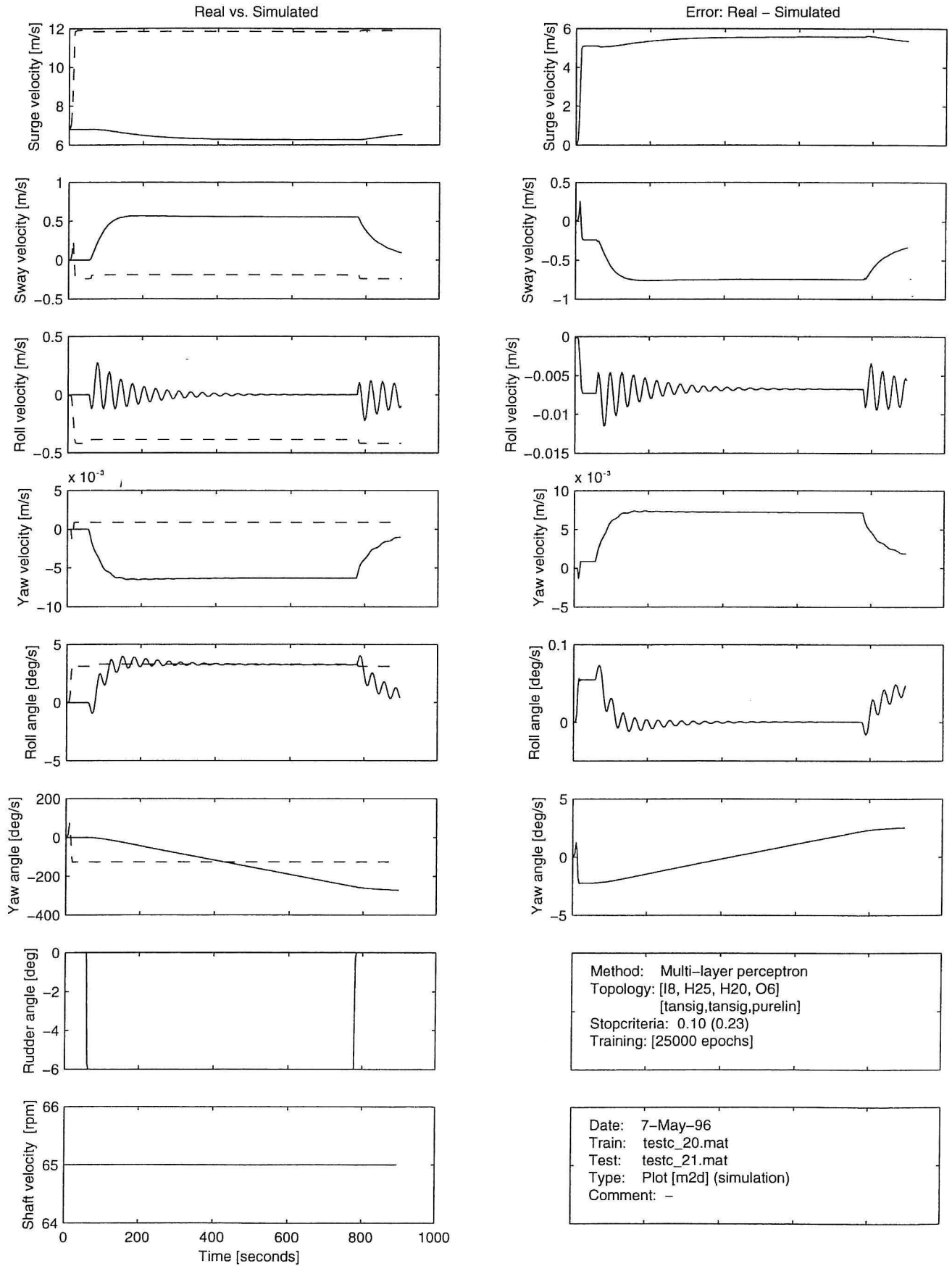

Real vs. Predicted
Error: Real – Predicted
Surge velocity [m/s]
Sway velocity [m/s]
Roll velocity [m/s]
Yaw velocity [m/s]
Roll angle [deg/s]
Yaw angle [deg/s]
Rudder angle [deg]
Shaft velocity [rpm]
x 10^3
x 10^4
Time [seconds]
Method: Multi–layer perceptron
Topology: [I8, H25, H20, O6]
[tansig,tansig,purelin]
Stopcriteria: 0.10 (0.22)
Training: [25000 epochs]
Date: 7–May–96
Train: testc_10.mat
Test: testc_11.mat
Type: Plot [m1c] (prediction)
Comment: –

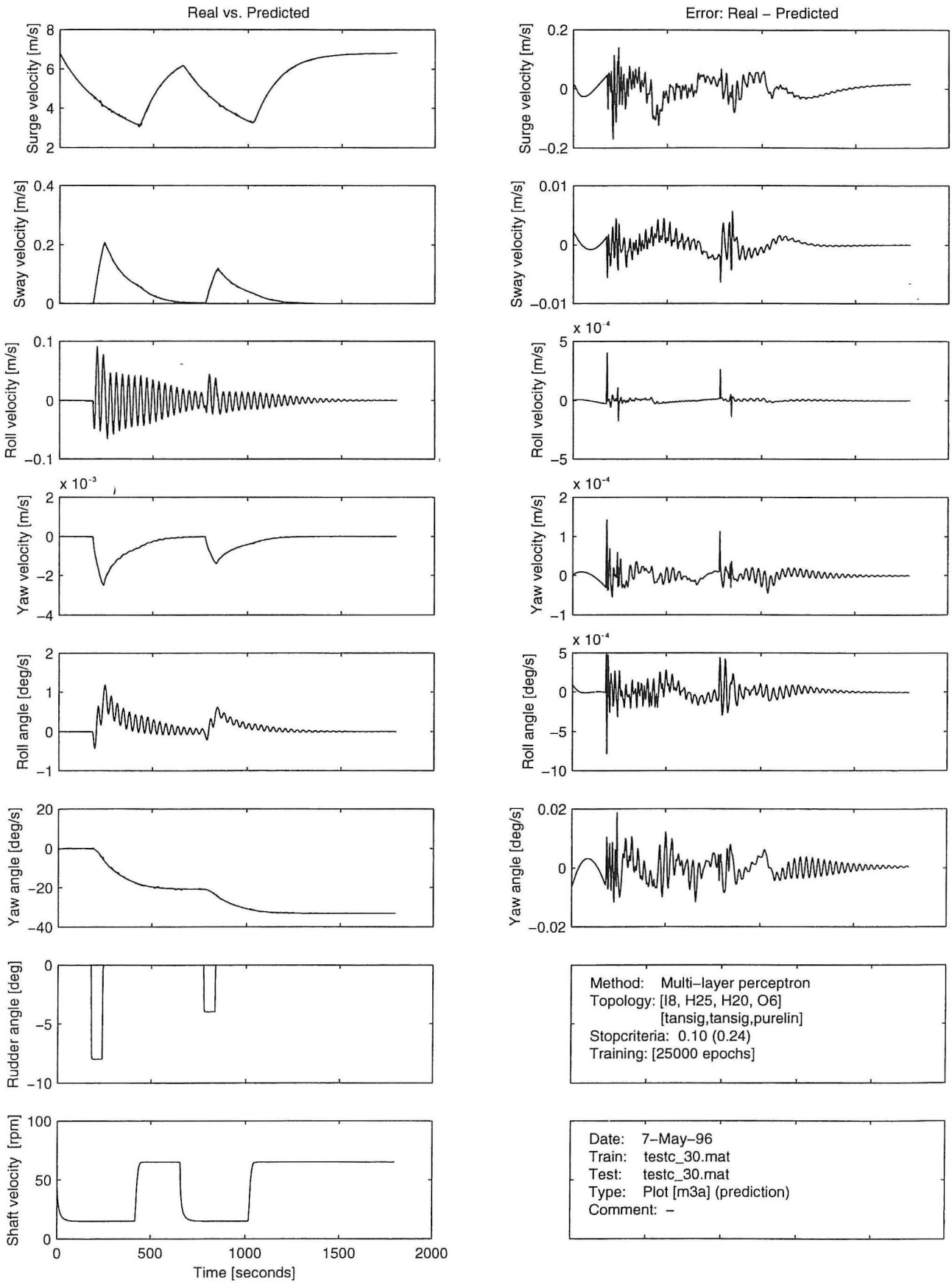

Real vs. Simulated
Error: Real – Simulated
Surge velocity [m/s]
Sway velocity [m/s]
Roll velocity [m/s]
Yaw velocity [m/s]
Roll angle [deg/s]
Yaw angle [deg/s]
Rudder angle [deg]
Shaft velocity [rpm]
x 10 ³
Time [seconds]
Method: Multi–layer perceptron
Topology: [I8, H25, H20, O6]
[tansig,tansig,purelin]
Stopcriteria: 0.10 (0.22)
Training: [25000 epochs]
Date: 7–May–96
Train: testc_10.mat
Test: testc_11.mat
Type: Plot [m1d] (simulation)
Comment: –

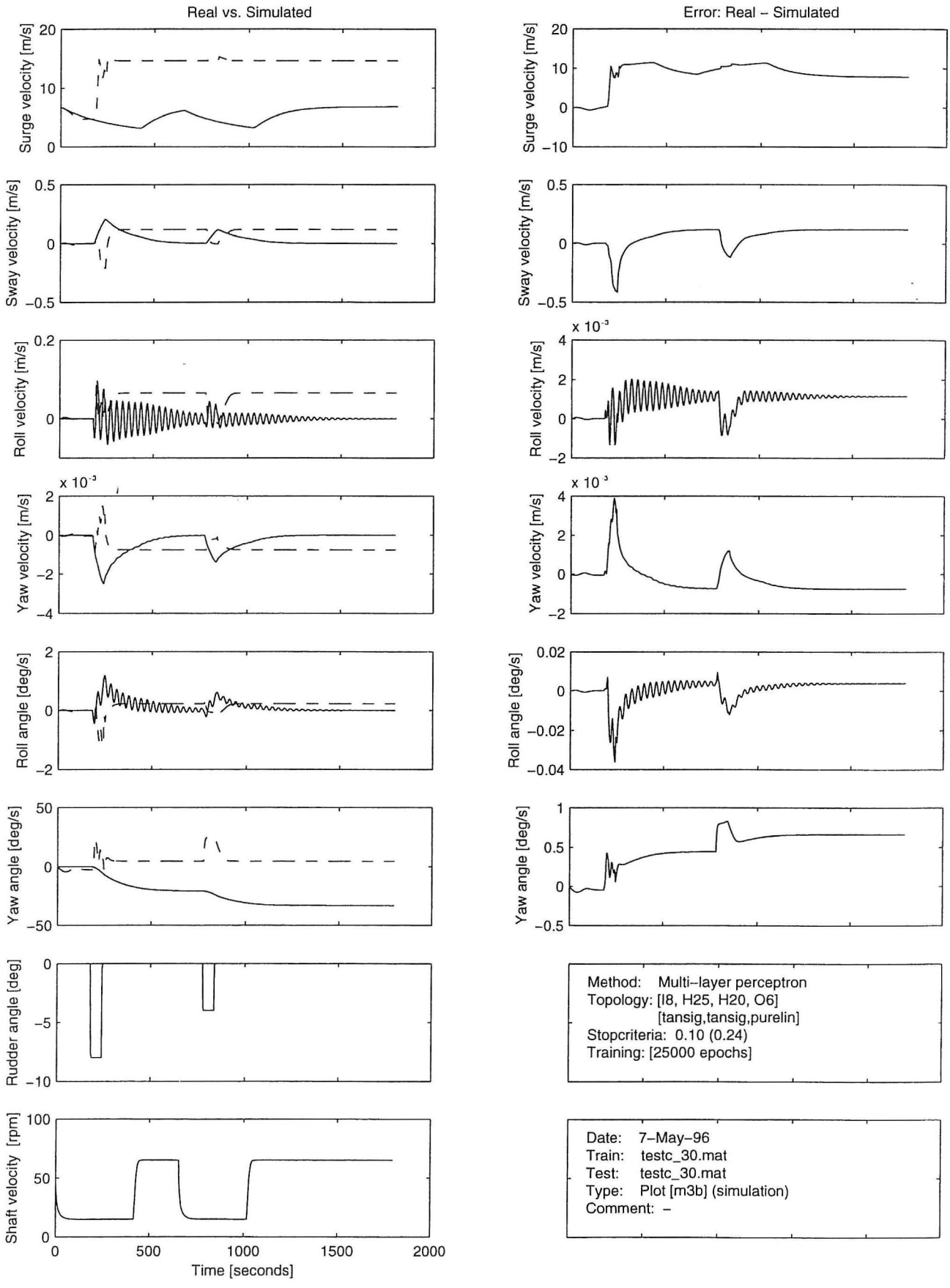

Real vs. Predicted
Error: Real – Predicted
Surge velocity [m/s]
Sway velocity [m/s]
Roll velocity [m/s]
Yaw velocity [m/s]
Roll angle [deg/s]
Yaw angle [deg/s]
Rudder angle [deg]
Shaft velocity [rpm]
Time [seconds]
x 10^-3
x 10^-4
Method: Multi-layer perceptron
Topology: [I8, H25, H20, O6]
[tansig,tansig,purelin]
Stopcriteria: 0.10 (0.23)
Training: [25000 epochs]
Date: 7-May-96
Train: testc_20.mat
Test: testc_20.mat
Type: Plot [m2a] (prediction)
Comment: –

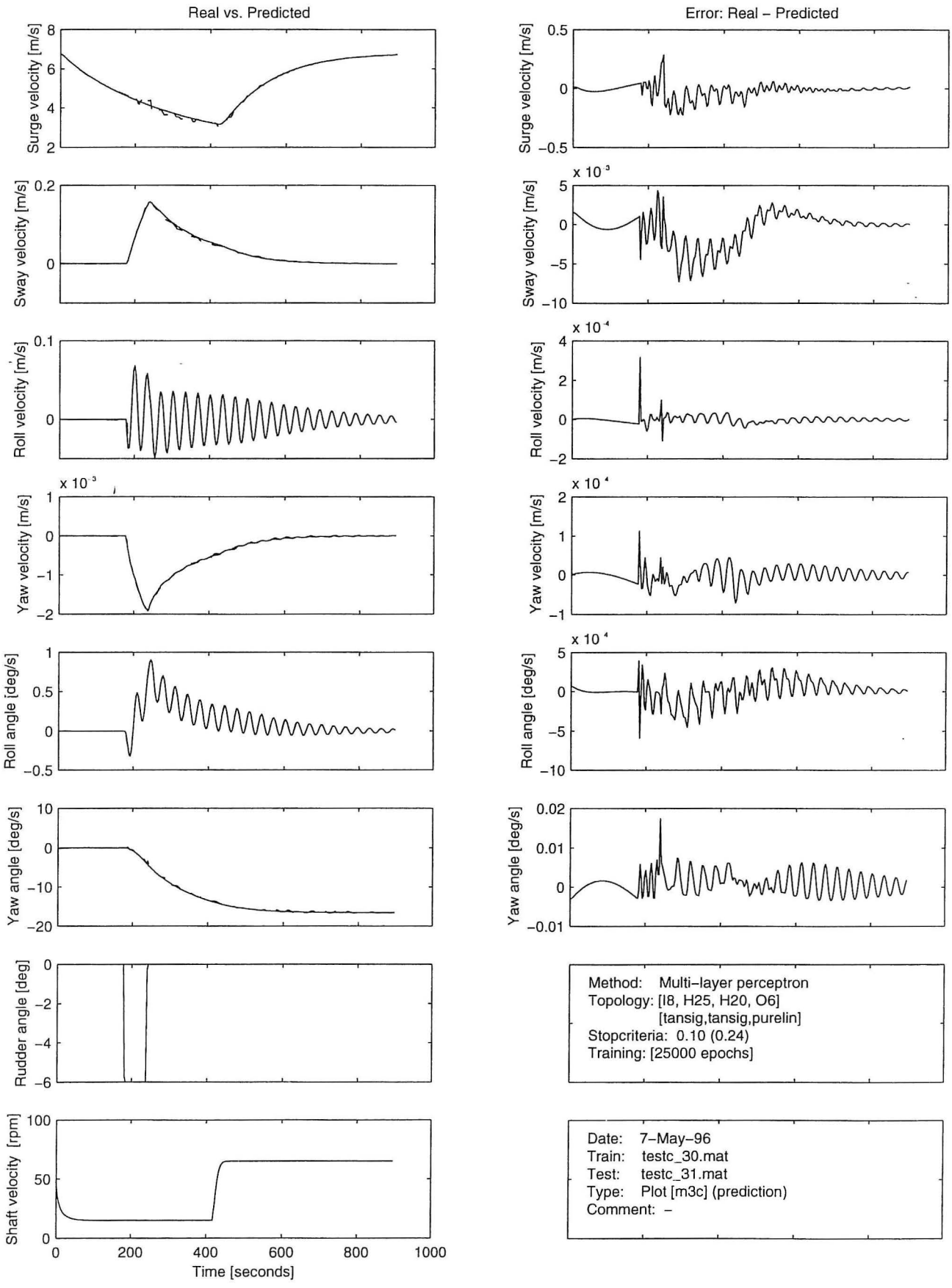

Real vs. Simulated
Error: Real – Simulated
Surge velocity [m/s]
Sway velocity [m/s]
Roll velocity [m/s]
Yaw velocity [m/s]
x 10^3
Roll angle [deg/s]
Yaw angle [deg/s]
Rudder angle [deg]
Shaft velocity [rpm]
Time [seconds]
Method: Multi–layer perceptron
Topology: [I8, H25, H20, O6]
[tansig,tansig,purelin]
Stopcriteria: 0.10 (0.23)
Training: [25000 epochs]
Date: 7–May–96
Train: testc_20.mat
Test: testc_20.mat
Type: Plot [m2b] (simulation)
Comment: –

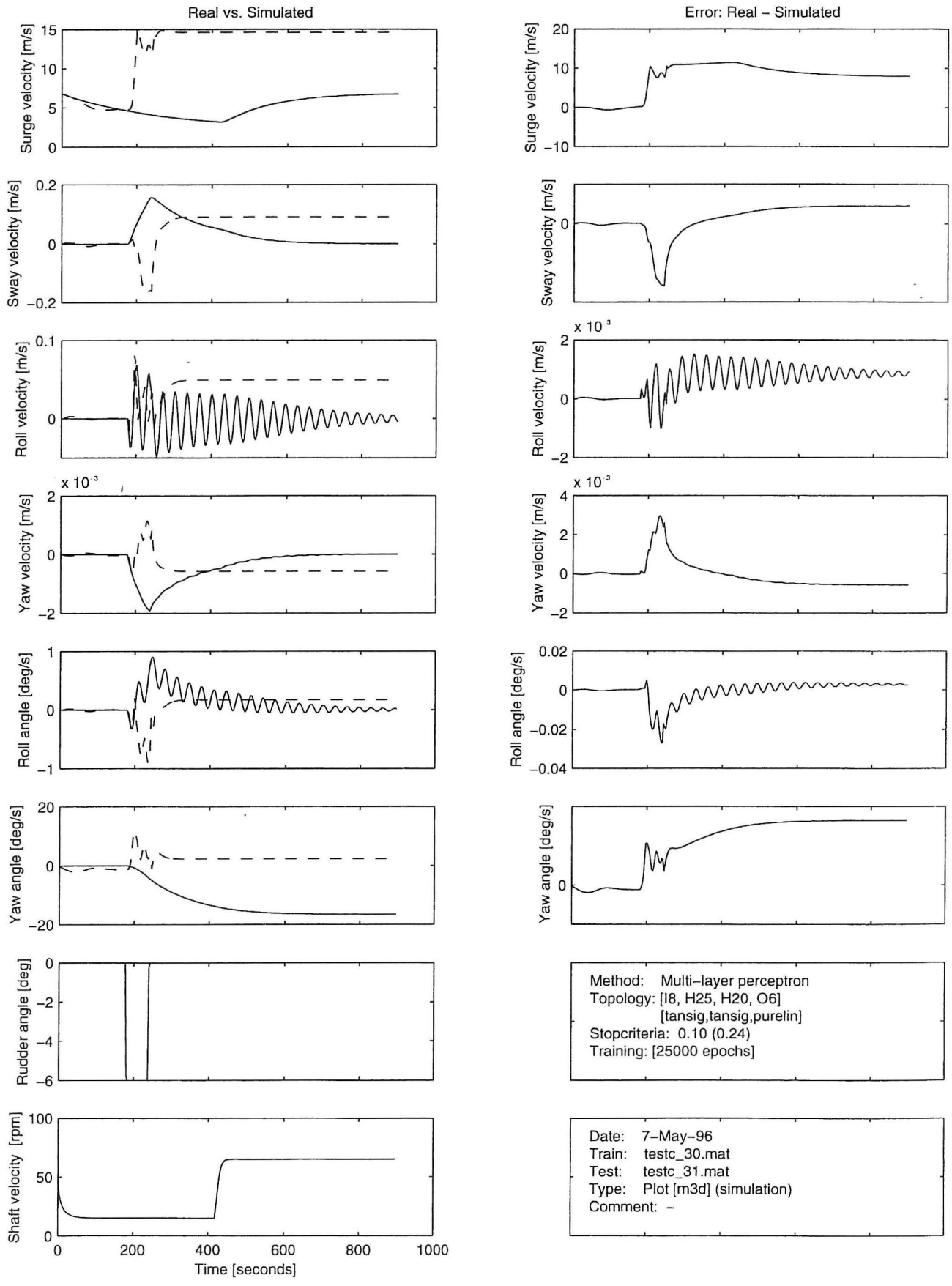

Real vs. Predicted
Error: Real – Predicted
Surge velocity [m/s]
Sway velocity [m/s]
Roll velocity [m/s]
Yaw velocity [m/s]
Roll angle [deg/s]
Yaw angle [deg/s]
Rudder angle [deg]
Shaft velocity [rpm]
x 10^-3
x 10^-4
Time [seconds]
Method: Multi-layer perceptron
Topology: [I8, H25, H20, O6]
[tansig,tansig,purelin]
Stopcriteria: 0.10 (0.23)
Training: [25000 epochs]
Date: 7-May-96
Train: testc_20.mat
Test: testc_21.mat
Type: Plot [m2c] (prediction)
Comment: –

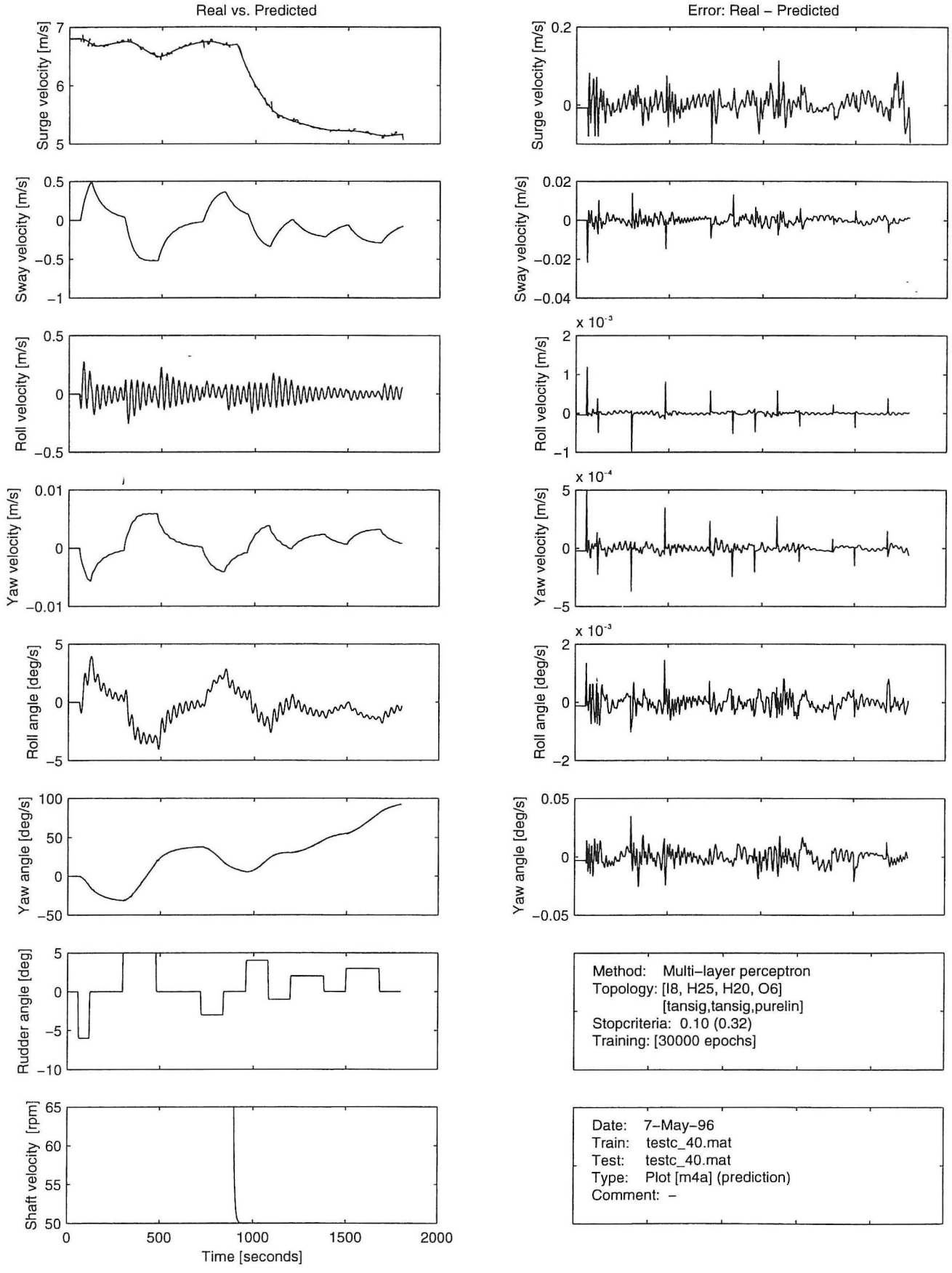

Real vs. Simulated
Error: Real – Simulated
Surge velocity [m/s]
Sway velocity [m/s]
Roll velocity [m/s]
Yaw velocity [m/s]
x 10⁻³
Roll angle [deg/s]
Yaw angle [deg/s]
Rudder angle [deg]
Shaft velocity [rpm]
Time [seconds]
Method: Multi-layer perceptron
Topology: [I8, H25, H20, O6]
[tansig,tansig,purelin]
Stopcriteria: 0.10 (0.23)
Training: [25000 epochs]
Date: 7-May-96
Train: testc_20.mat
Test: testc_21.mat
Type: Plot [m2d] (simulation)
Comment: –

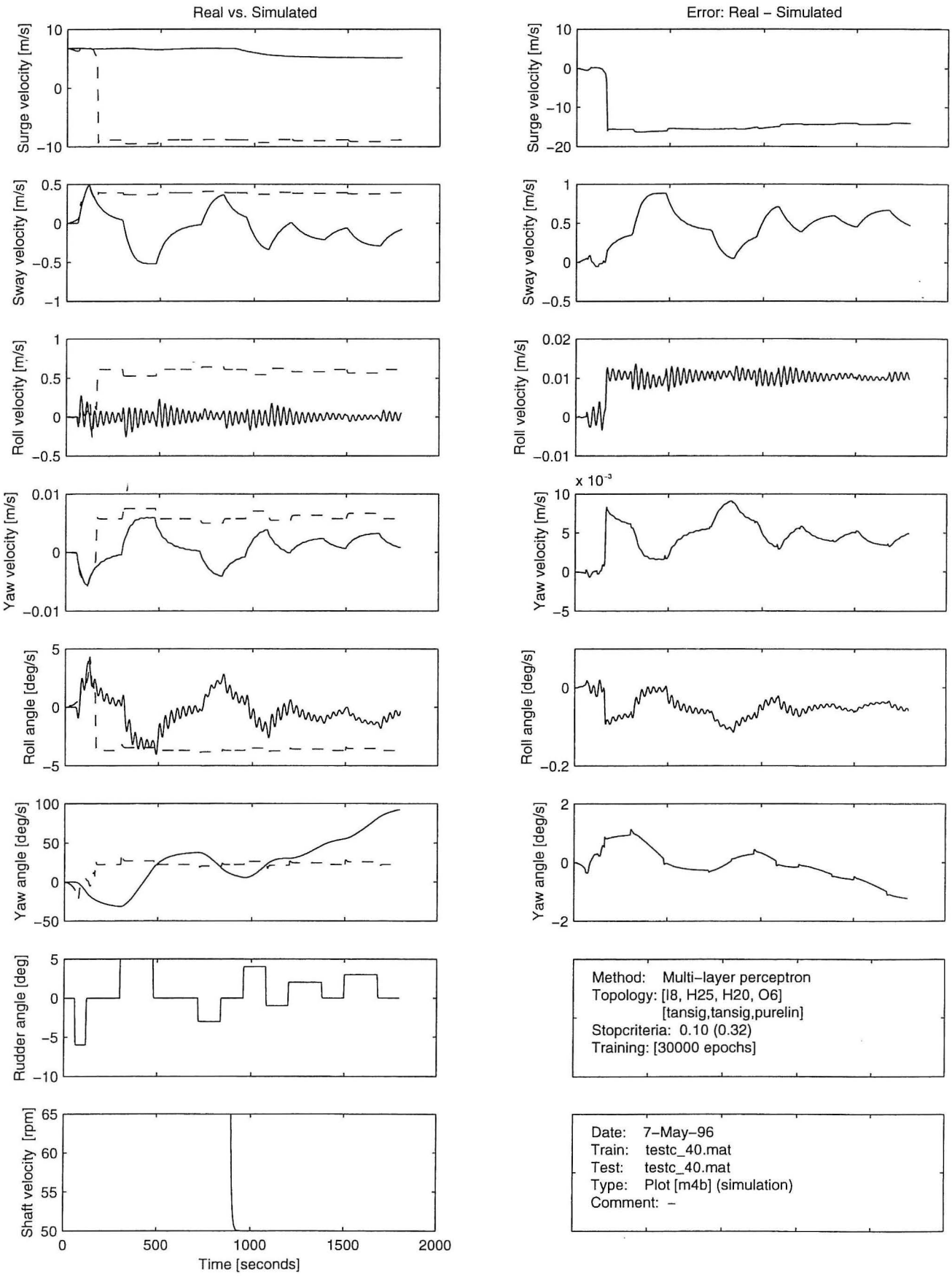

Real vs. Predicted
Error: Real – Predicted
Surge velocity [m/s]
Sway velocity [m/s]
Roll velocity [m/s]
Yaw velocity [m/s]
Roll angle [deg/s]
Yaw angle [deg/s]
Rudder angle [deg]
Shaft velocity [rpm]
x 10^-3
x 10^-4
Time [seconds]
Method: Multi-layer perceptron
Topology: [I8, H25, H20, O6]
[tansig,tansig,purelin]
Stopcriteria: 0.10 (0.24)
Training: [25000 epochs]
Date: 7-May-96
Train: testc_30.mat
Test: testc_30.mat
Type: Plot [m3a] (prediction)
Comment: –

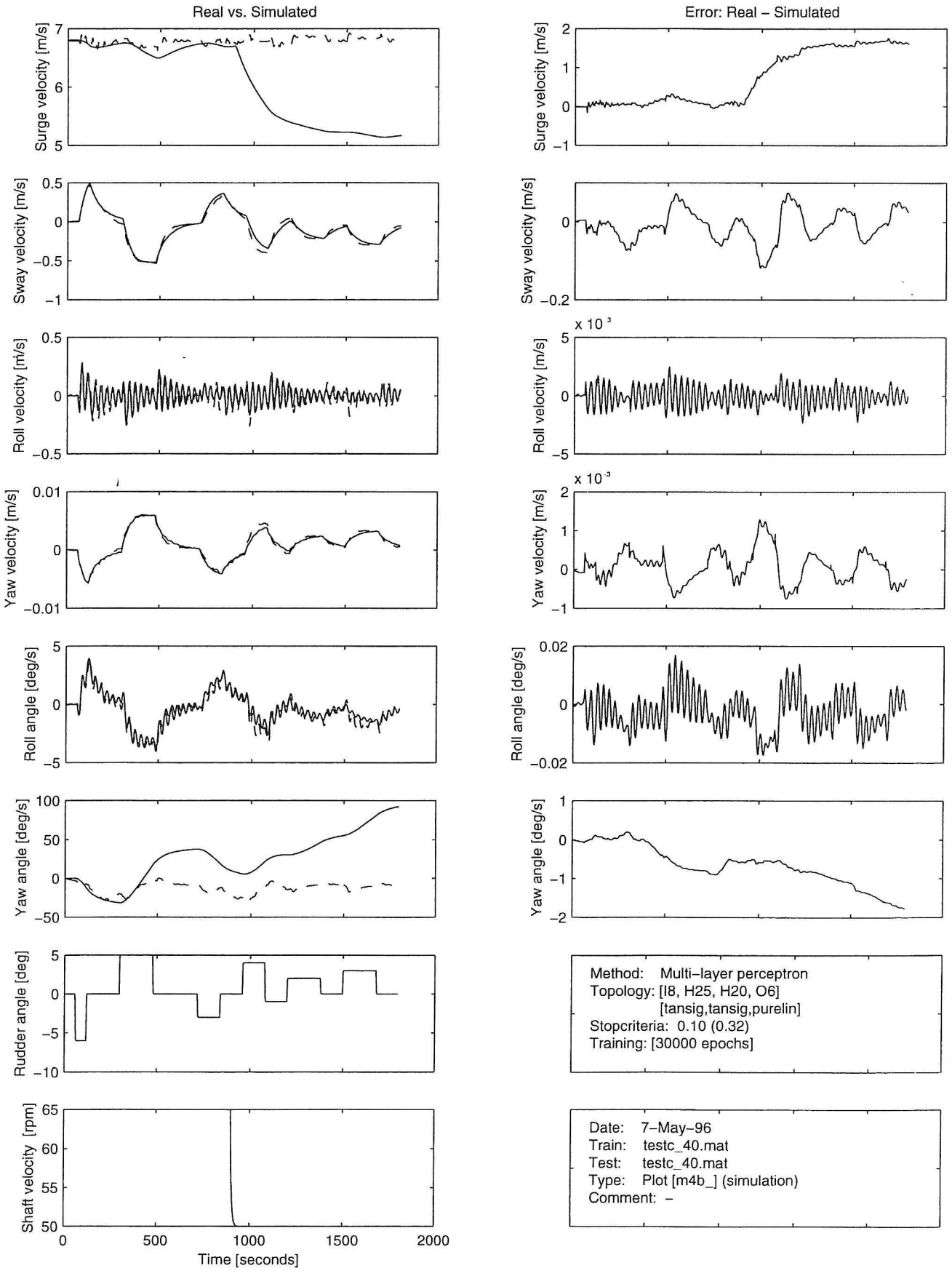

Real vs. Simulated
Error: Real – Simulated
Surge velocity [m/s]
Sway velocity [m/s]
Roll velocity [m/s]
Yaw velocity [m/s]
Roll angle [deg/s]
Yaw angle [deg/s]
Rudder angle [deg]
Shaft velocity [rpm]
x 10⁻³
Time [seconds]
Method: Multi–layer perceptron
Topology: [I8, H25, H20, O6]
[tansig,tansig,purelin]
Stopcriteria: 0.10 (0.24)
Training: [25000 epochs]
Date: 7–May–96
Train: testc_30.mat
Test: testc_30.mat
Type: Plot [m3b] (simulation)
Comment: –

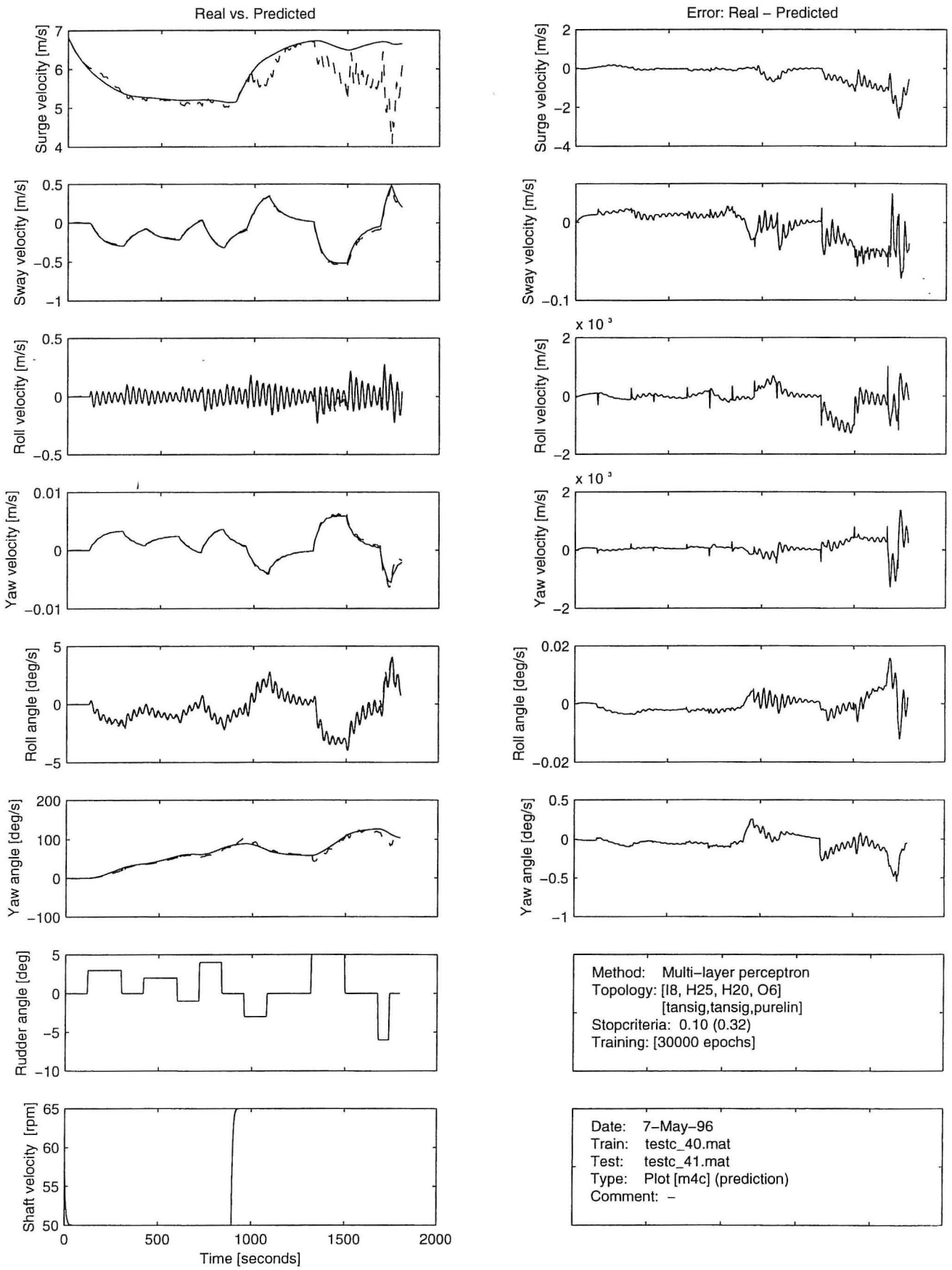

Real vs. Predicted
Error: Real – Predicted
Surge velocity [m/s]
Sway velocity [m/s]
Roll velocity [m/s]
Yaw velocity [m/s]
Roll angle [deg/s]
Yaw angle [deg/s]
Rudder angle [deg]
Shaft velocity [rpm]
x 10^-3
x 10^-4
x 10^4
Time [seconds]
Method: Multi-layer perceptron
Topology: [I8, H25, H20, O6]
[tansig,tansig,purelin]
Stopcriteria: 0.10 (0.24)
Training: [25000 epochs]
Date: 7-May-96
Train: testc_30.mat
Test: testc_31.mat
Type: Plot [m3c] (prediction)
Comment: –

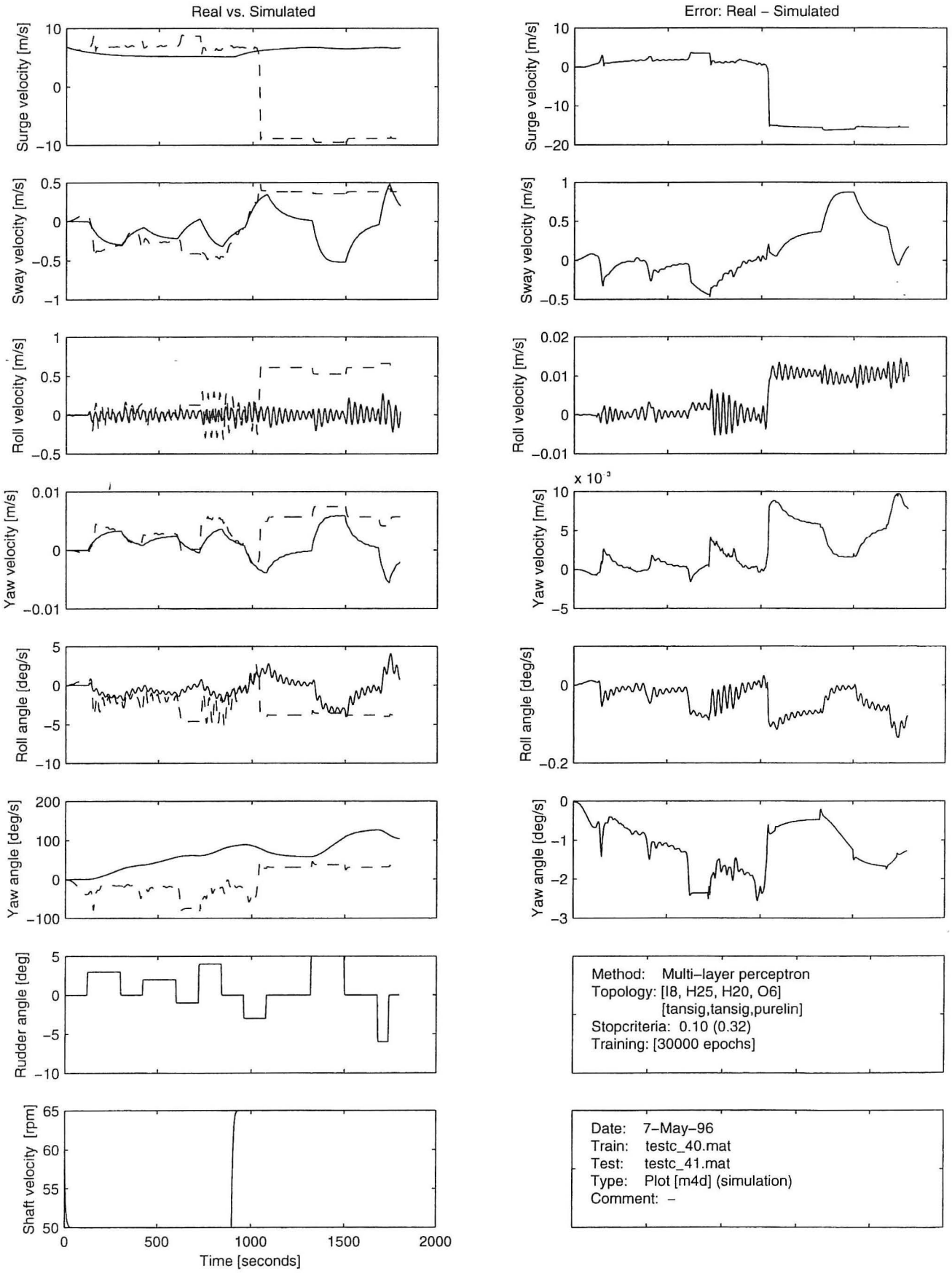

Real vs. Simulated
Error: Real – Simulated
Surge velocity [m/s]
Sway velocity [m/s]
Roll velocity [m/s]
Yaw velocity [m/s]
Roll angle [deg/s]
Yaw angle [deg/s]
Rudder angle [deg]
Shaft velocity [rpm]
x 10⁻³
Time [seconds]
Method: Multi-layer perceptron
Topology: [I8, H25, H20, O6]
[tansig,tansig,purelin]
Stopcriteria: 0.10 (0.24)
Training: [25000 epochs]
Date: 7-May-96
Train: testc_30.mat
Test: testc_31.mat
Type: Plot [m3d] (simulation)
Comment: -

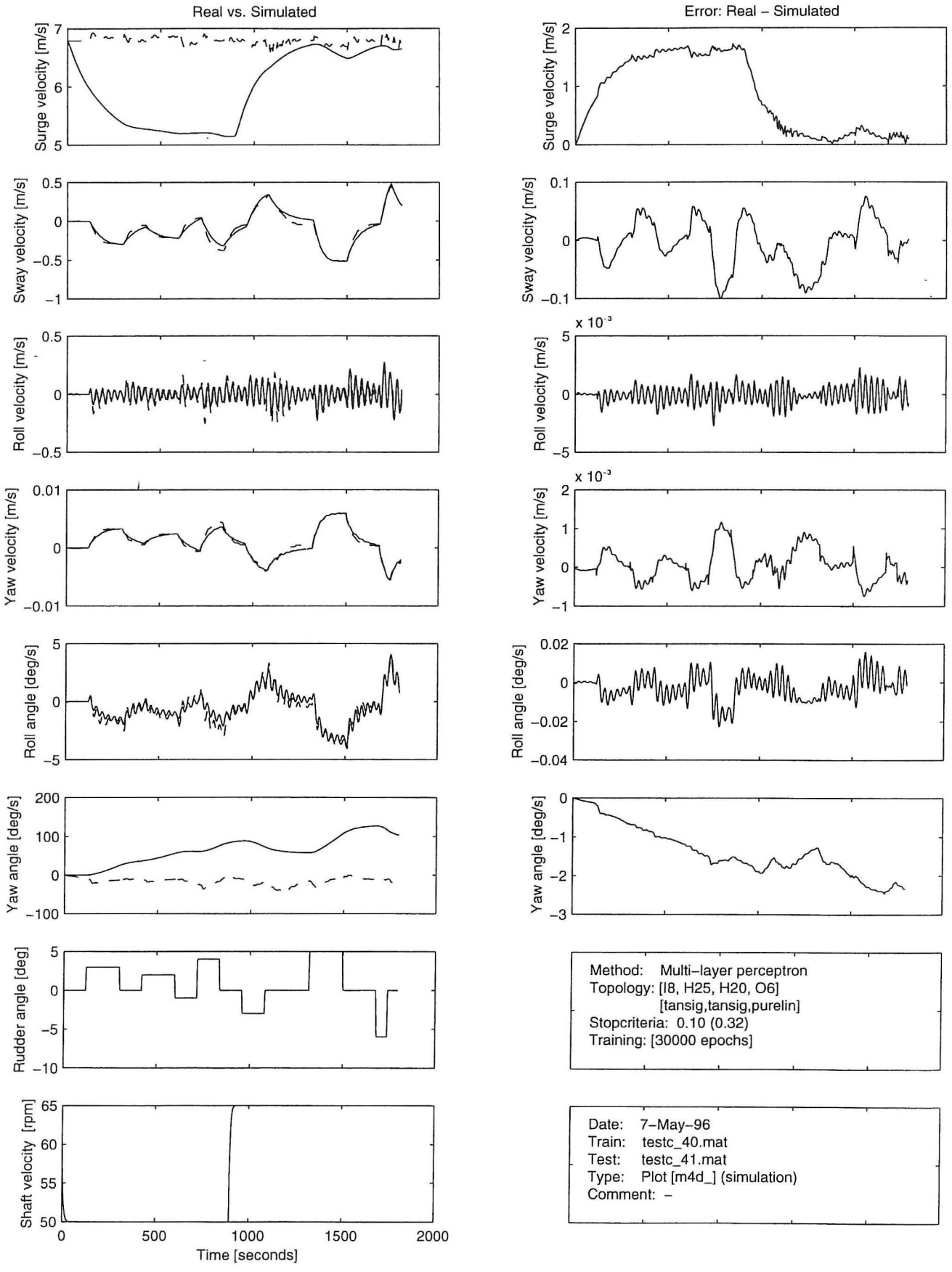

Real vs. Predicted
Error: Real – Predicted
Surge velocity [m/s]
Sway velocity [m/s]
Roll velocity [m/s]
Yaw velocity [m/s]
Roll angle [deg/s]
Yaw angle [deg/s]
Rudder angle [deg]
Shaft velocity [rpm]
Time [seconds]
x 10^-3
x 10^-4
Method: Multi-layer perceptron
Topology: [I8, H25, H20, O6]
[tansig,tansig,purelin]
Stopcriteria: 0.10 (0.32)
Training: [30000 epochs]
Date: 7-May-96
Train: testc_40.mat
Test: testc_40.mat
Type: Plot [m4a] (prediction)
Comment: –


Real vs. Simulated
Error: Real – Simulated
Surge velocity [m/s]
Sway velocity [m/s]
Roll velocity [m/s]
Yaw velocity [m/s]
Roll angle [deg/s]
Yaw angle [deg/s]
Rudder angle [deg]
Shaft velocity [rpm]
x 10^-3
Time [seconds]
Method: Multi-layer perceptron
Topology: [I8, H25, H20, O6]
[tansig,tansig,purelin]
Stopcriteria: 0.10 (0.32)
Training: [30000 epochs]
Date: 7-May-96
Train: testc_40.mat
Test: testc_40.mat
Type: Plot [m4b] (simulation)
Comment: –

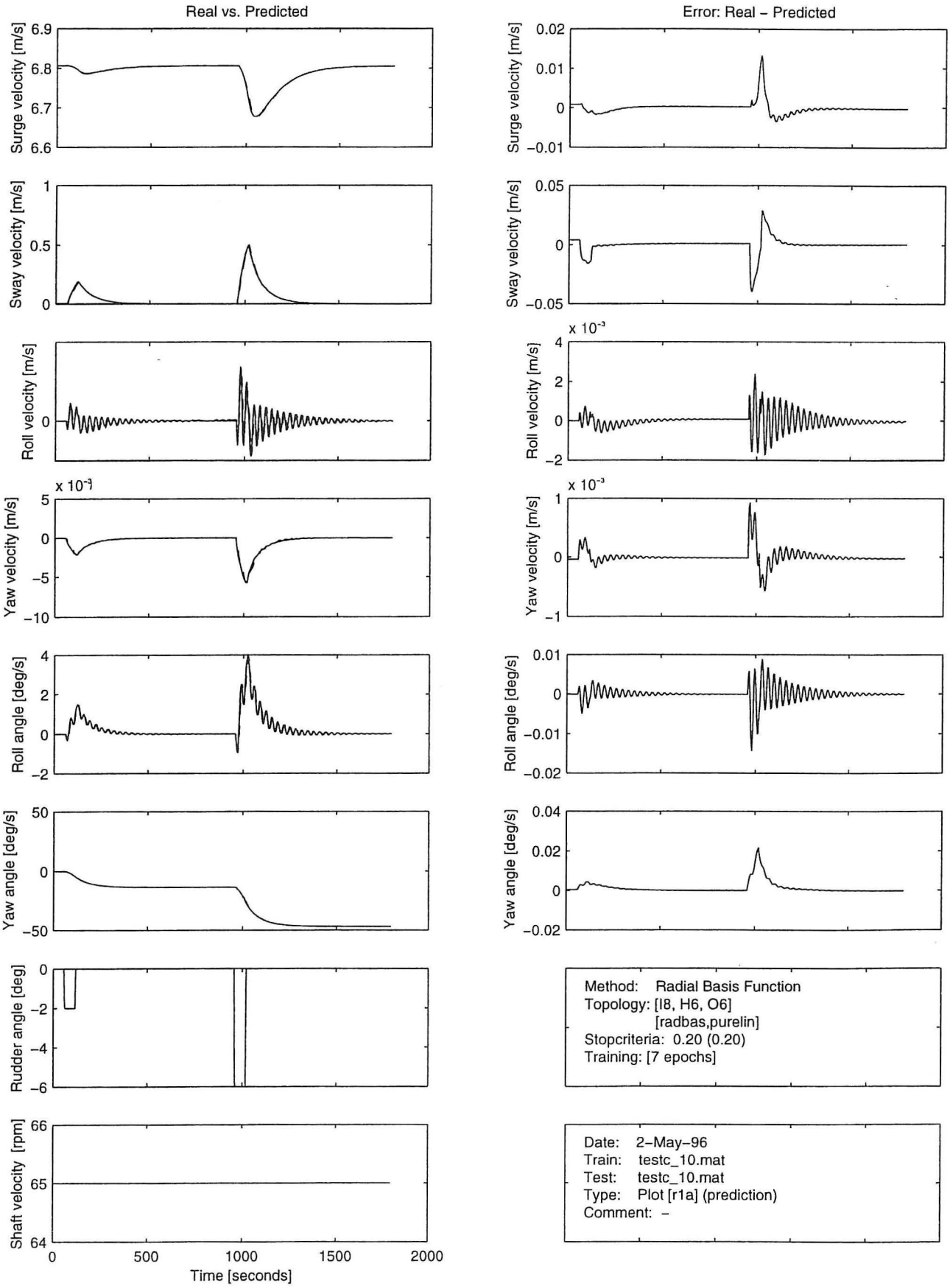

Real vs. Simulated
Error: Real – Simulated
Surge velocity [m/s]
Sway velocity [m/s]
Roll velocity [m/s]
Yaw velocity [m/s]
Roll angle [deg/s]
Yaw angle [deg/s]
Rudder angle [deg]
Shaft velocity [rpm]
x 10^-3
Time [seconds]
Method: Multi-layer perceptron
Topology: [I8, H25, H20, O6]
[tansig,tansig,purelin]
Stopcriteria: 0.10 (0.32)
Training: [30000 epochs]
Date: 7-May-96
Train: testc_40.mat
Test: testc_40.mat
Type: Plot [m4b_] (simulation)
Comment: –

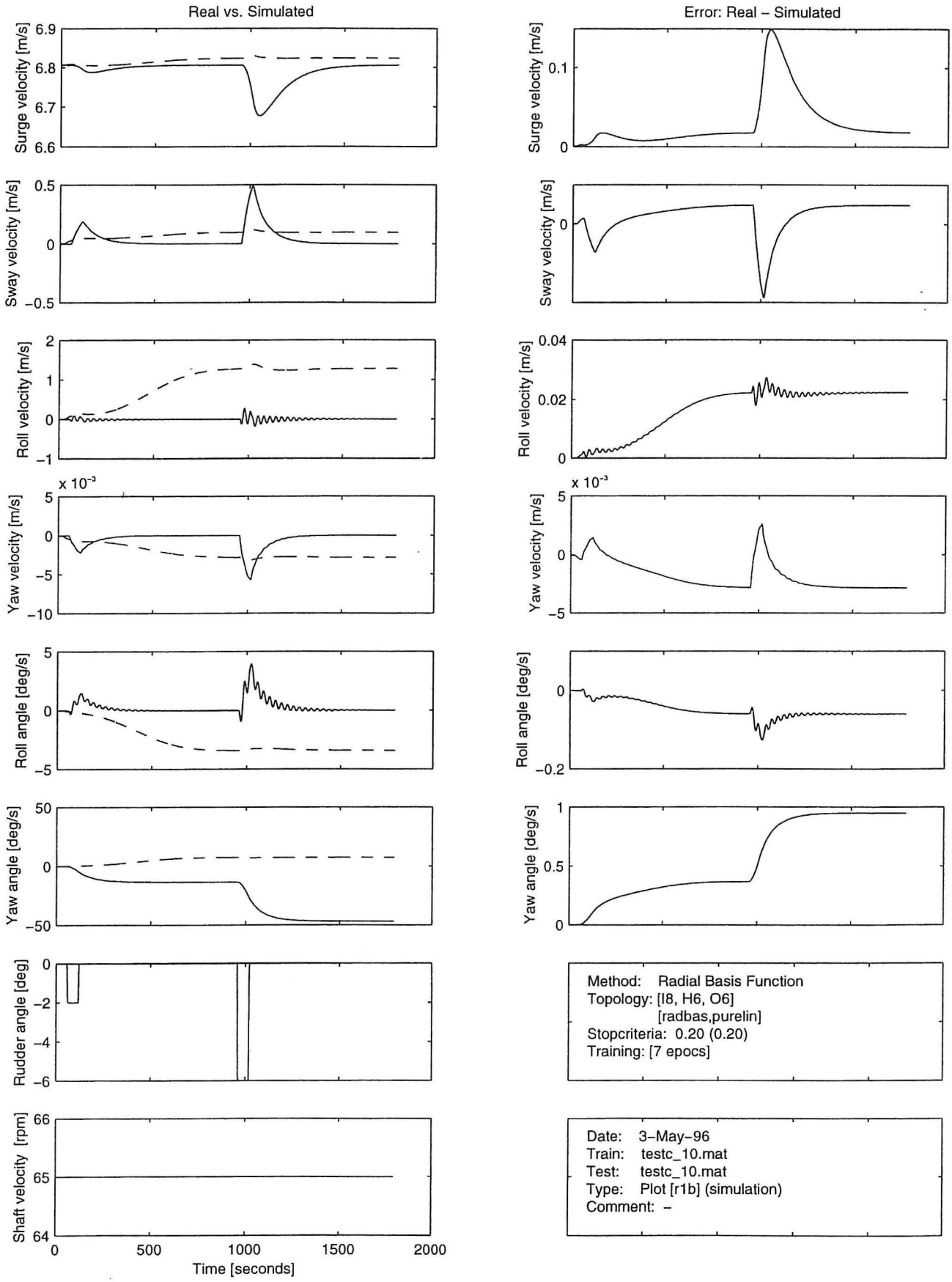

Real vs. Predicted
Error: Real – Predicted
Surge velocity [m/s]
Sway velocity [m/s]
Roll velocity [m/s]
Yaw velocity [m/s]
Roll angle [deg/s]
Yaw angle [deg/s]
Rudder angle [deg]
Shaft velocity [rpm]
x 10 ³
Time [seconds]
Method: Multi–layer perceptron
Topology: [I8, H25, H20, O6]
[tansig,tansig,purelin]
Stopcriteria: 0.10 (0.32)
Training: [30000 epochs]
Date: 7–May–96
Train: testc_40.mat
Test: testc_41.mat
Type: Plot [m4c] (prediction)
Comment: –

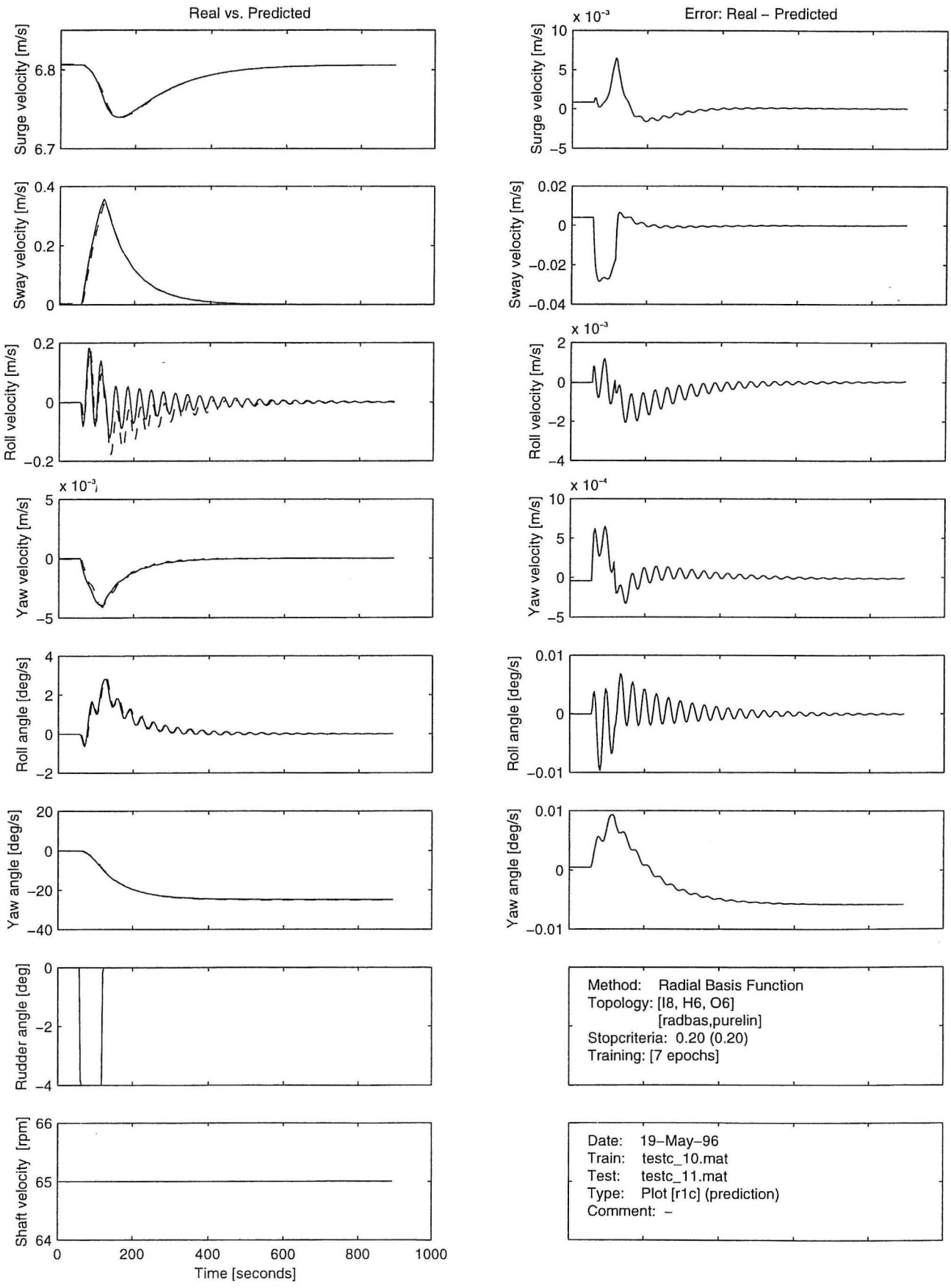

Real vs. Simulated
Error: Real – Simulated
Surge velocity [m/s]
Sway velocity [m/s]
Roll velocity [m/s]
Yaw velocity [m/s]
Roll angle [deg/s]
Yaw angle [deg/s]
Rudder angle [deg]
Shaft velocity [rpm]
x 10^-3
Time [seconds]
Method: Multi–layer perceptron
Topology: [I8, H25, H20, O6]
[tansig,tansig,purelin]
Stopcriteria: 0.10 (0.32)
Training: [30000 epochs]
Date: 7–May–96
Train: testc_40.mat
Test: testc_41.mat
Type: Plot [m4d] (simulation)
Comment: –

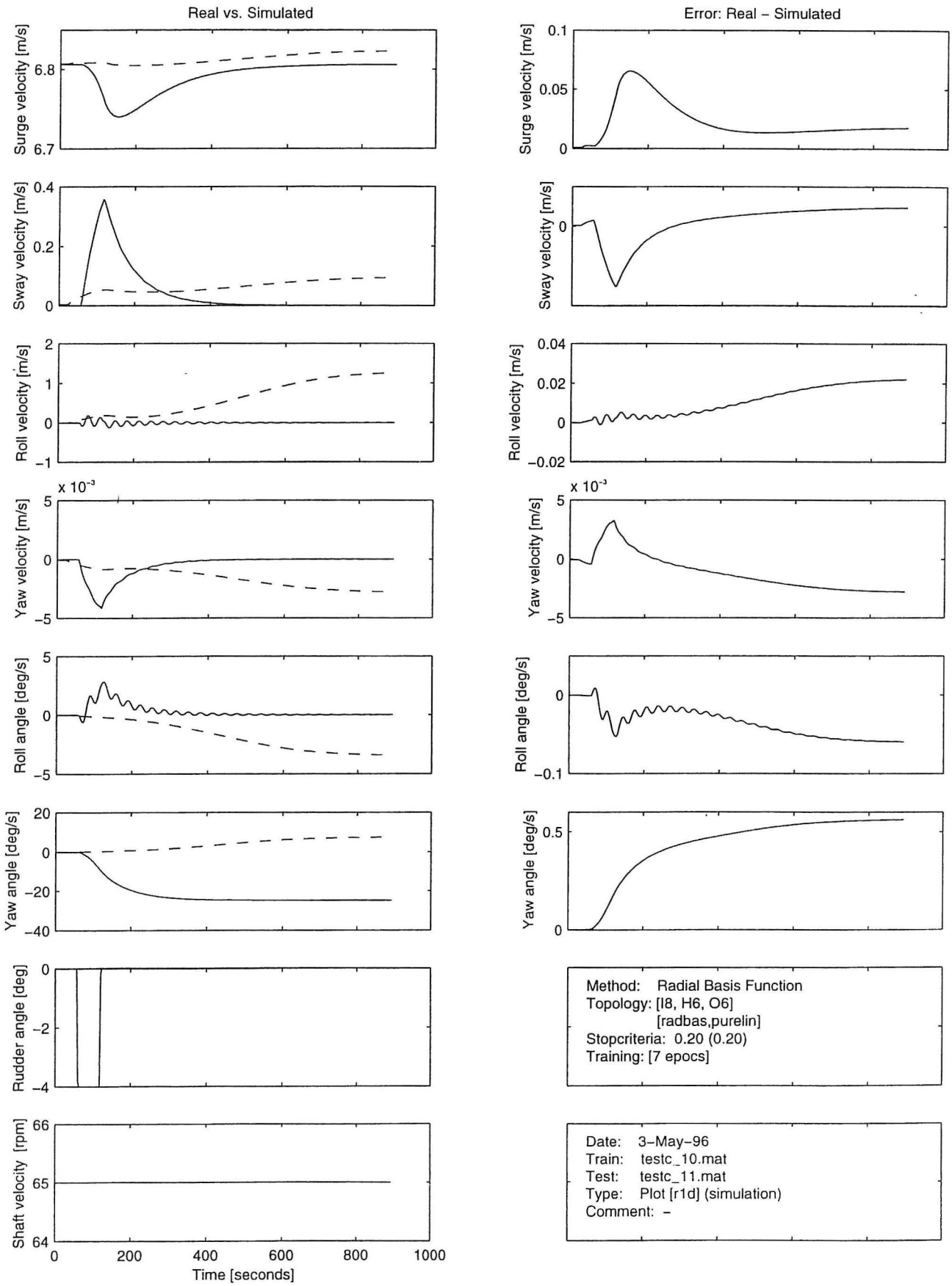

Real vs. Simulated
Error: Real – Simulated
Surge velocity [m/s]
Sway velocity [m/s]
Roll velocity [m/s]
Yaw velocity [m/s]
Roll angle [deg/s]
Yaw angle [deg/s]
Rudder angle [deg]
Shaft velocity [rpm]
x 10⁻³
Time [seconds]
Method: Multi-layer perceptron
Topology: [I8, H25, H20, O6]
[tansig,tansig,purelin]
Stopcriteria: 0.10 (0.32)
Training: [30000 epochs]
Date: 7-May-96
Train: testc_40.mat
Test: testc_41.mat
Type: Plot [m4d_] (simulation)
Comment: –

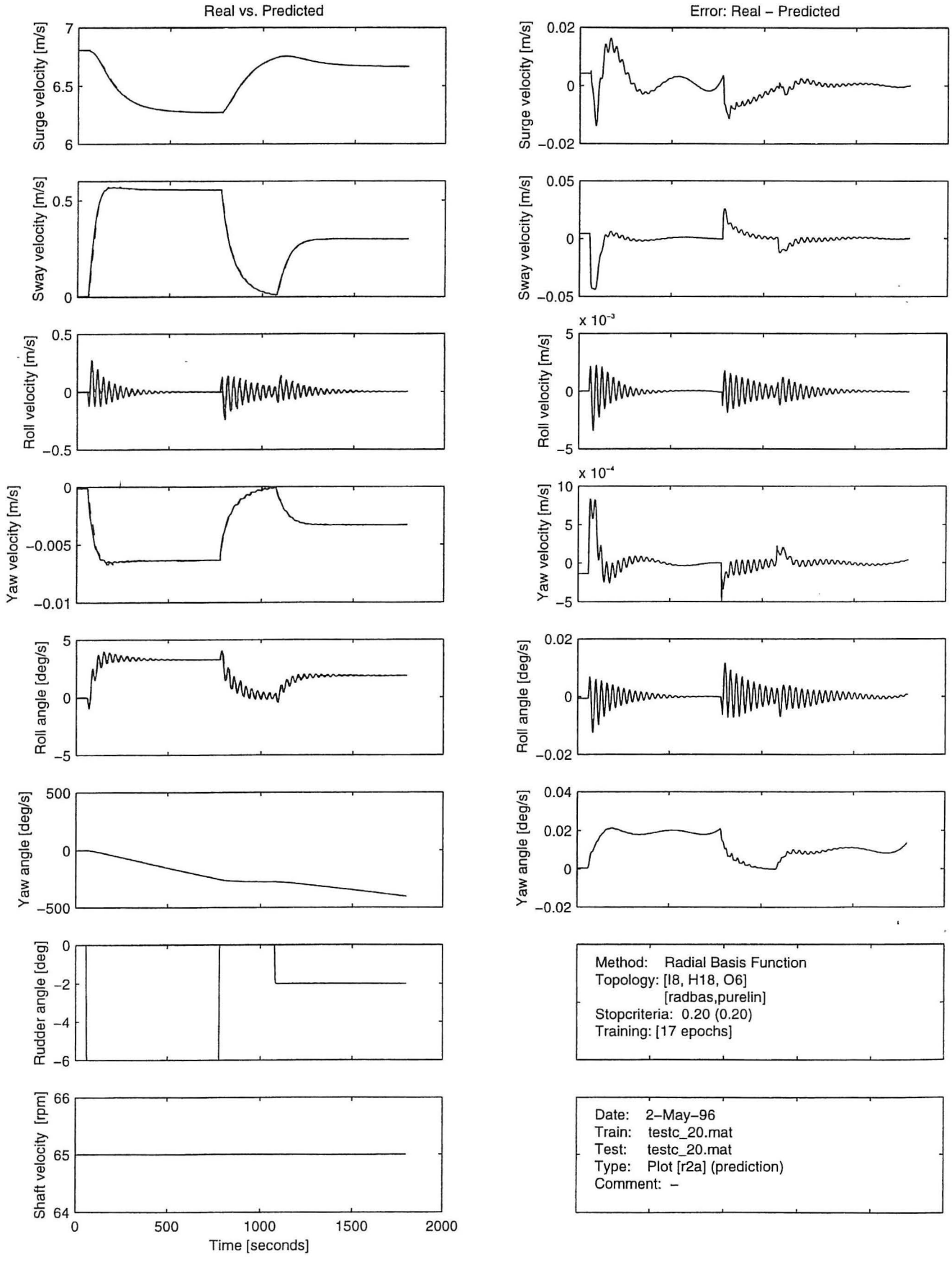
Appendix A.2 - RBFN Results

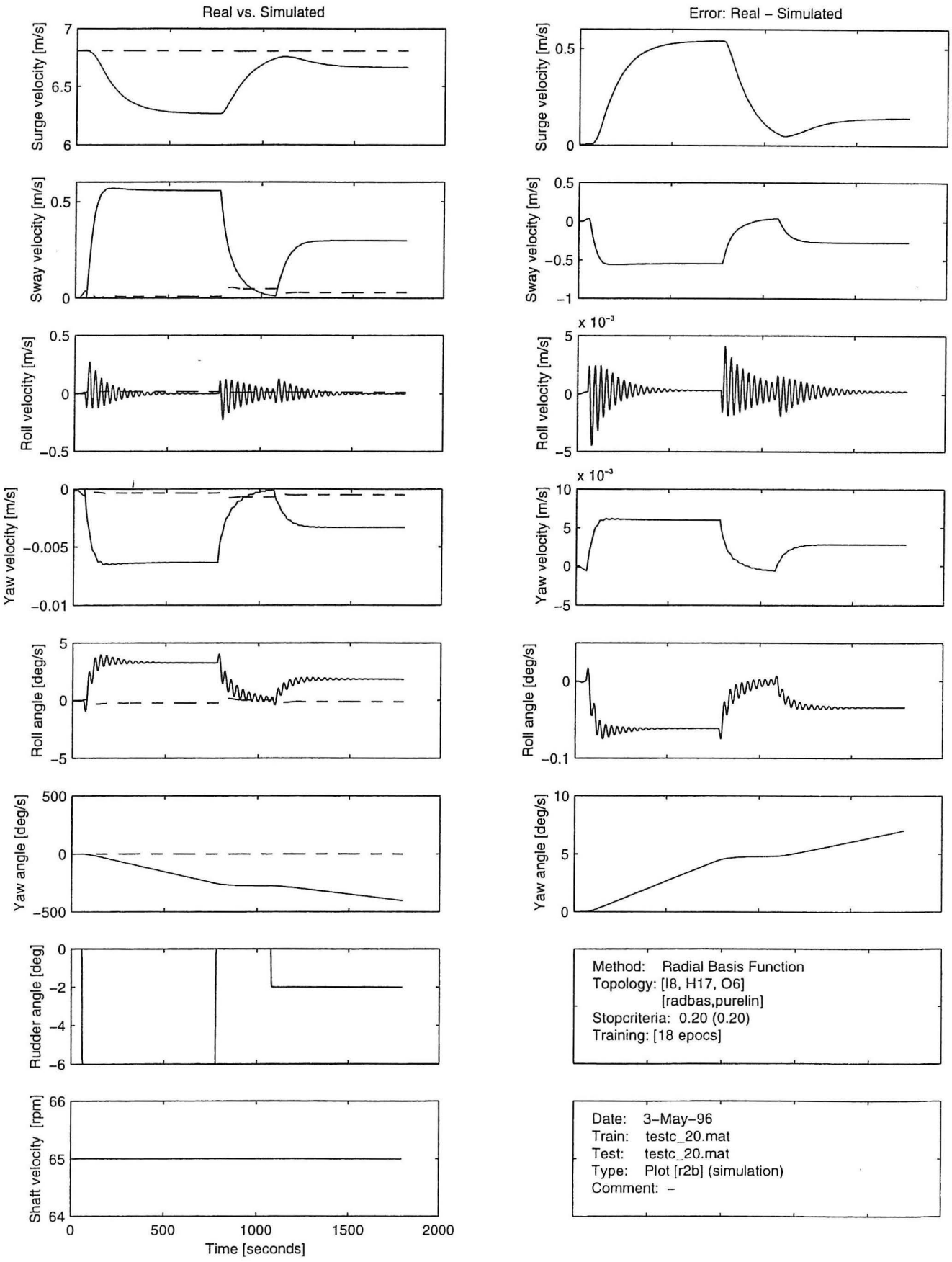


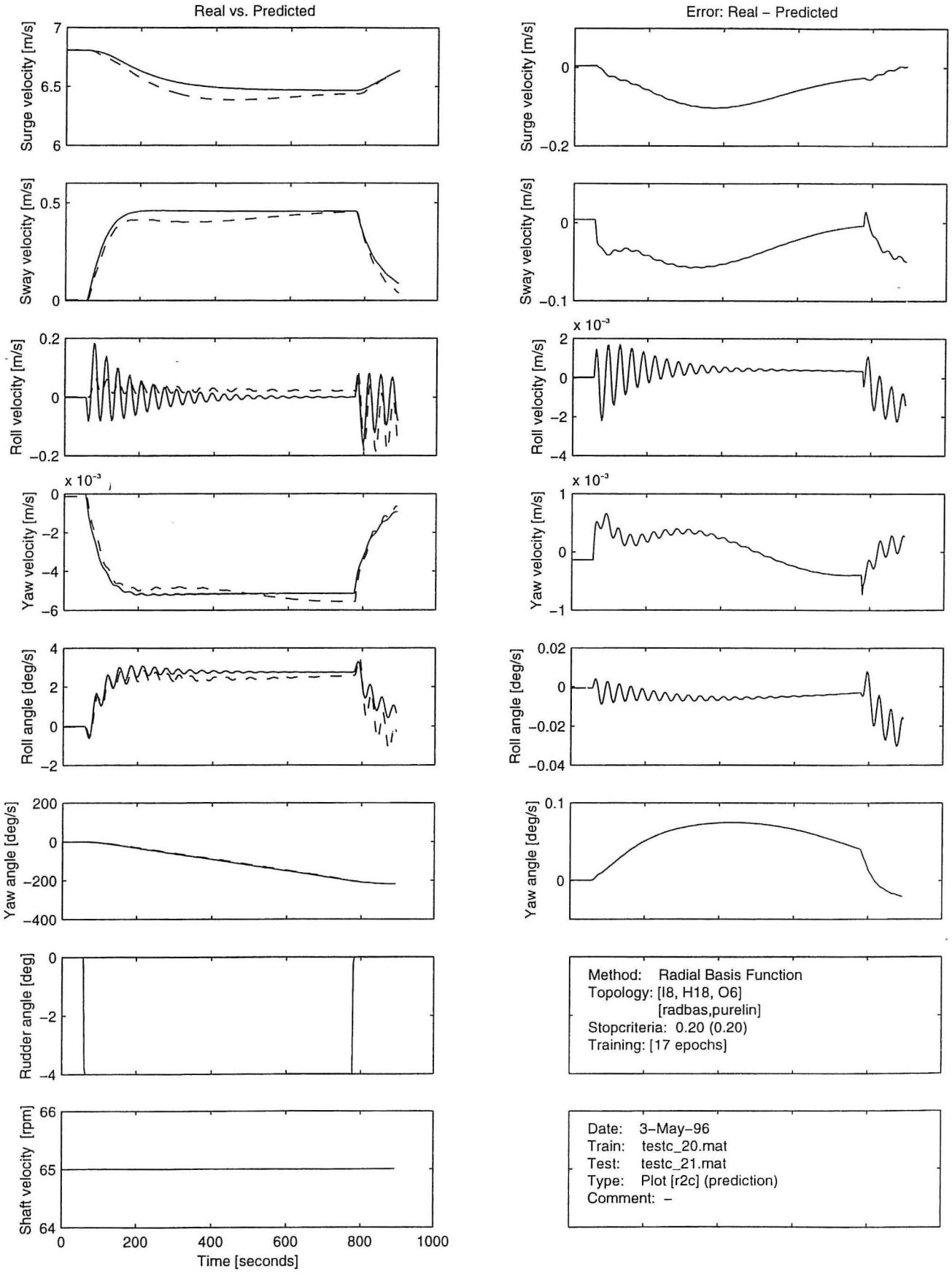

Real vs. Simulated
Error: Real – Simulated
Surge velocity [m/s]
Sway velocity [m/s]
Roll velocity [m/s]
Yaw velocity [m/s]
Roll angle [deg/s]
Yaw angle [deg/s]
Rudder angle [deg]
Shaft velocity [rpm]
x 10⁻³
Time [seconds]
Method: Radial Basis Function
Topology: [I8, H6, O6]
[radbas,purelin]
Stopcriteria: 0.20 (0.20)
Training: [7 epocs]
Date: 3-May-96
Train: testc_10.mat
Test: testc_10.mat
Type: Plot [r1b] (simulation)
Comment: –

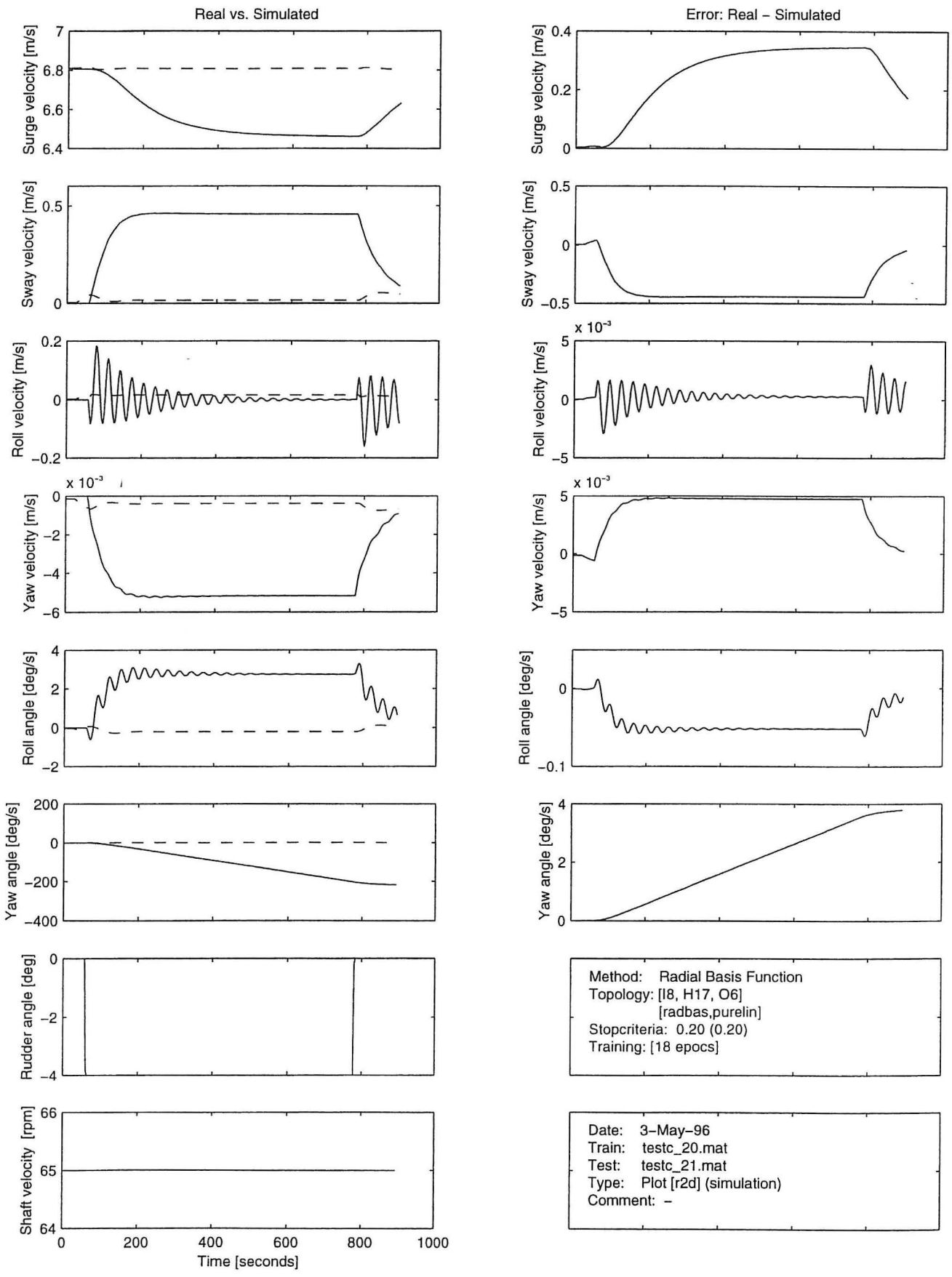

Real vs. Predicted
Error: Real – Predicted
Surge velocity [m/s]
Sway velocity [m/s]
Roll velocity [m/s]
Yaw velocity [m/s]
Roll angle [deg/s]
Yaw angle [deg/s]
Rudder angle [deg]
Shaft velocity [rpm]
x 10⁻³
x 10⁻⁴
Time [seconds]
Method: Radial Basis Function
Topology: [I8, H6, O6]
[radbas,purelin]
Stopcriteria: 0.20 (0.20)
Training: [7 epochs]
Date: 19–May–96
Train: testc_10.mat
Test: testc_11.mat
Type: Plot [r1c] (prediction)
Comment: –

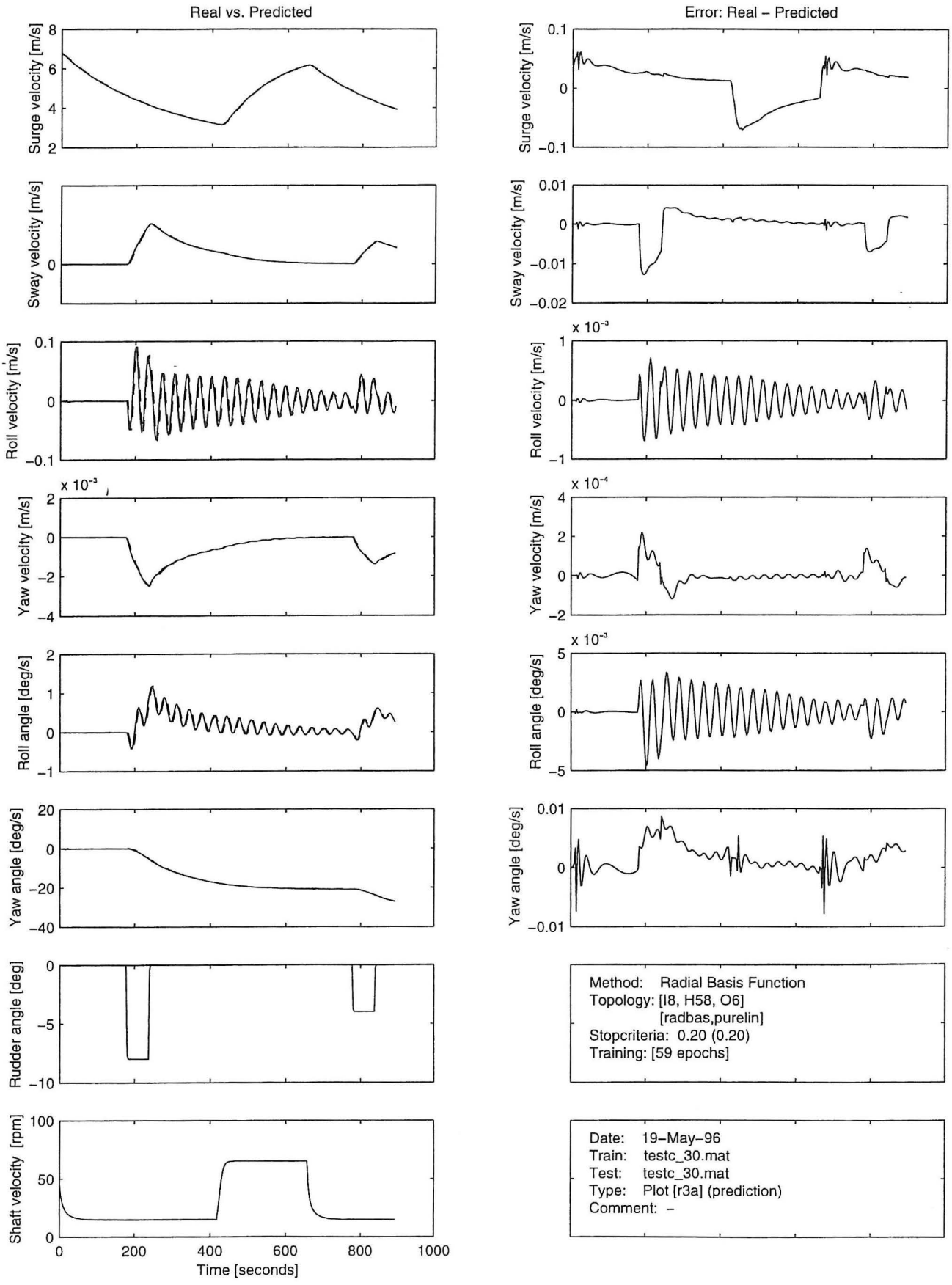

Real vs. Simulated
Error: Real – Simulated
Surge velocity [m/s]
Sway velocity [m/s]
Roll velocity [m/s]
Yaw velocity [m/s]
Roll angle [deg/s]
Yaw angle [deg/s]
Rudder angle [deg]
Shaft velocity [rpm]
x 10⁻³
Time [seconds]
Method: Radial Basis Function
Topology: [I8, H6, O6]
[radbas,purelin]
Stopcriteria: 0.20 (0.20)
Training: [7 epocs]
Date: 3-May-96
Train: testc_10.mat
Test: testc_11.mat
Type: Plot [r1d] (simulation)
Comment: –

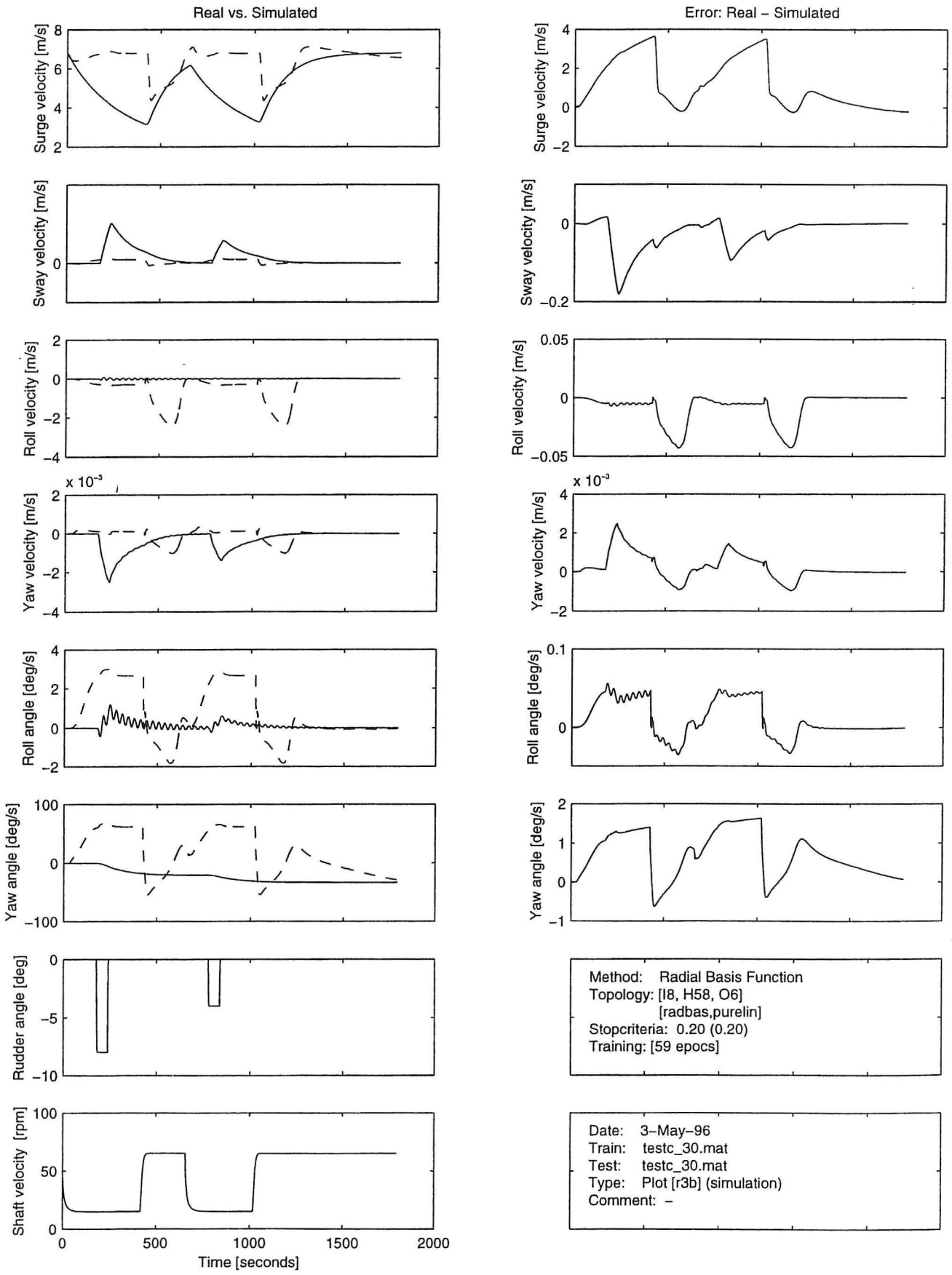

Real vs. Predicted
Error: Real – Predicted
Surge velocity [m/s]
Sway velocity [m/s]
Roll velocity [m/s]
Yaw velocity [m/s]
Roll angle [deg/s]
Yaw angle [deg/s]
Rudder angle [deg]
Shaft velocity [rpm]
x 10^-3
x 10^-4
Time [seconds]
Method: Radial Basis Function
Topology: [I8, H18, O6]
[radbas,purelin]
Stopcriteria: 0.20 (0.20)
Training: [17 epochs]
Date: 2–May–96
Train: testc_20.mat
Test: testc_20.mat
Type: Plot [r2a] (prediction)
Comment: –

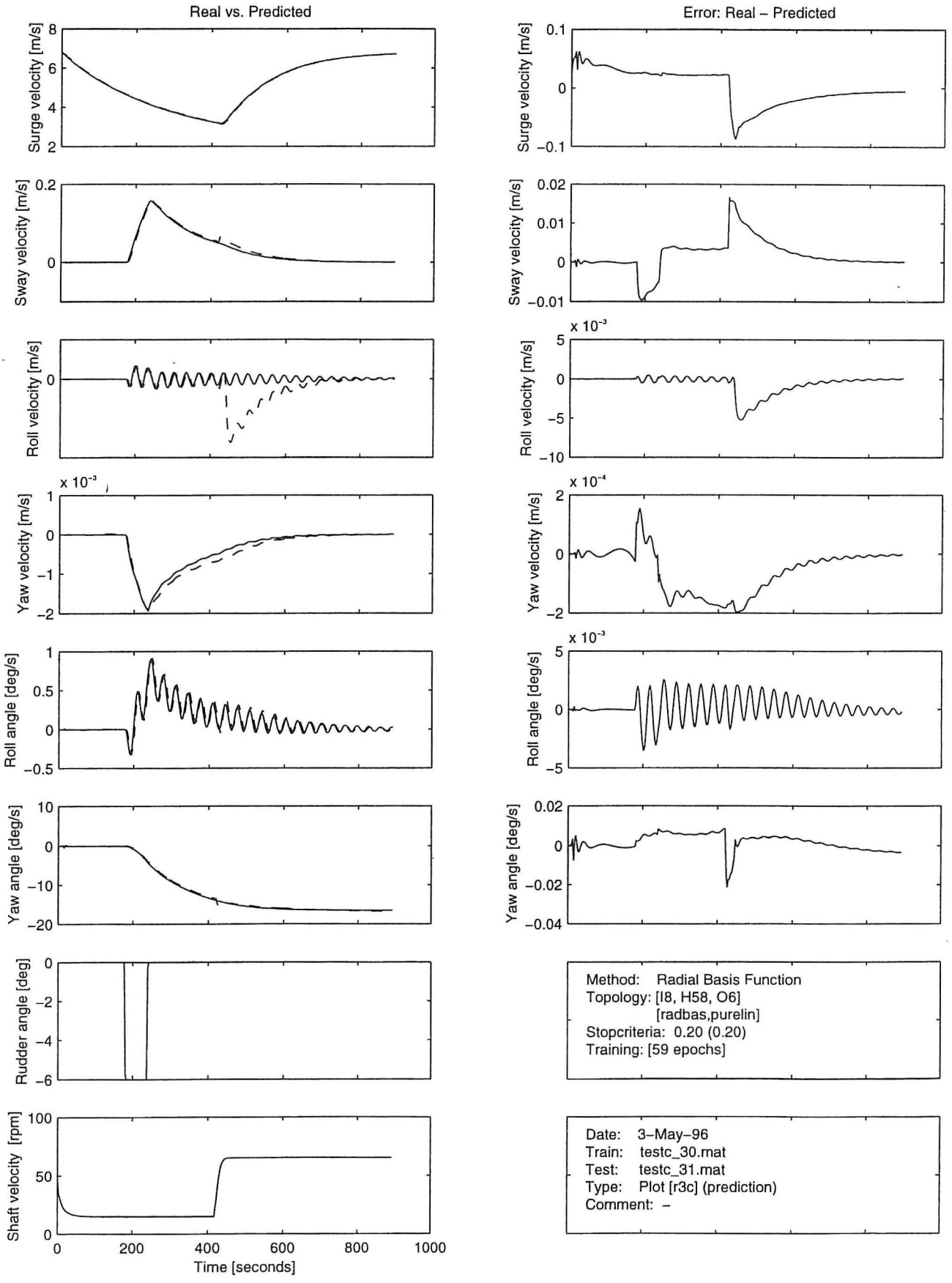

Real vs. Simulated
Error: Real – Simulated
Surge velocity [m/s]
Sway velocity [m/s]
Roll velocity [m/s]
Yaw velocity [m/s]
Roll angle [deg/s]
Yaw angle [deg/s]
Rudder angle [deg]
Shaft velocity [rpm]
x 10⁻³
Time [seconds]
Method: Radial Basis Function
Topology: [I8, H17, O6]
[radbas,purelin]
Stopcriteria: 0.20 (0.20)
Training: [18 epocs]
Date: 3-May-96
Train: testc_20.mat
Test: testc_20.mat
Type: Plot [r2b] (simulation)
Comment: –

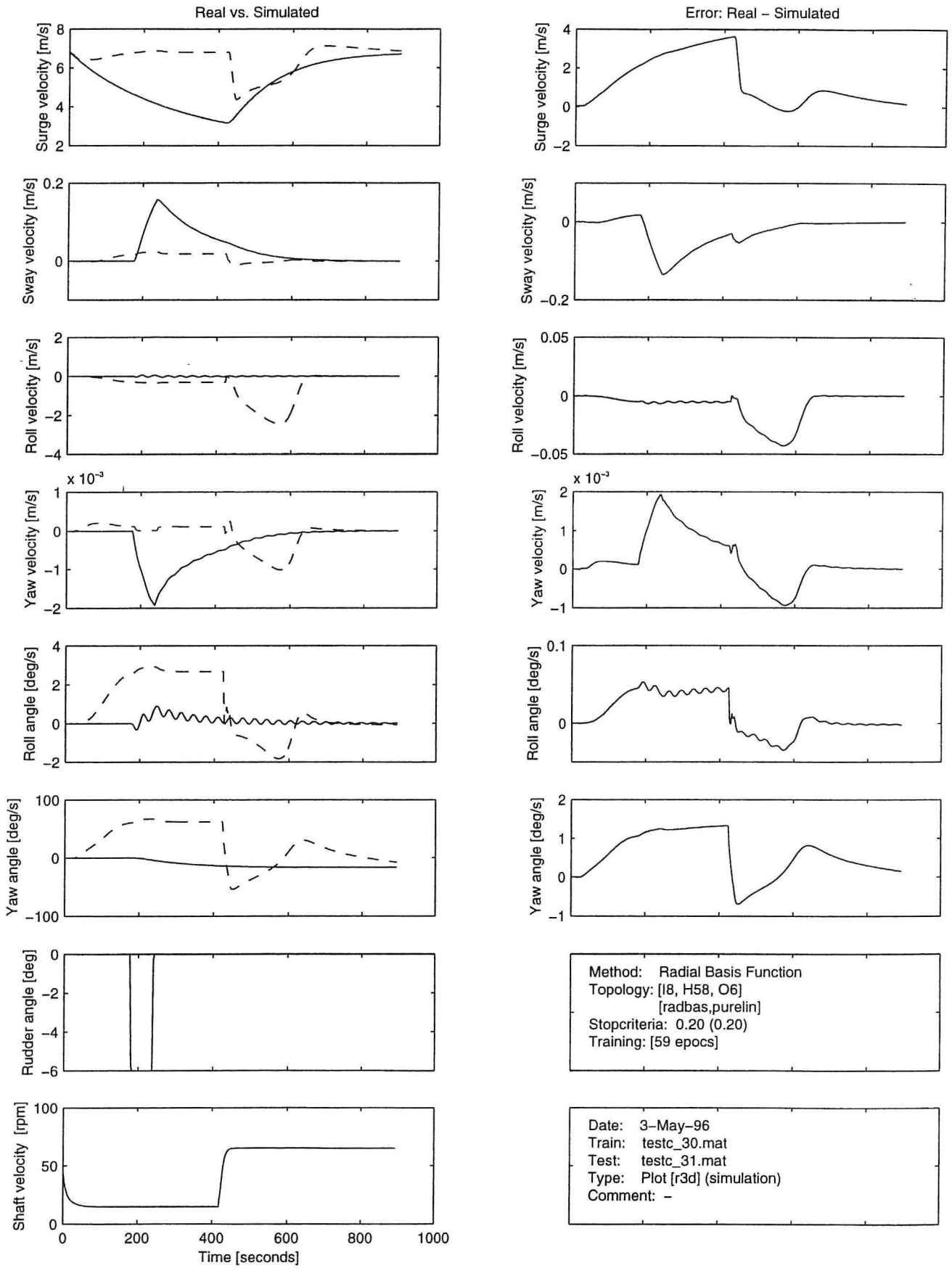

Real vs. Predicted
Error: Real – Predicted
Surge velocity [m/s]
Sway velocity [m/s]
Roll velocity [m/s]
Yaw velocity [m/s]
Roll angle [deg/s]
Yaw angle [deg/s]
Rudder angle [deg]
Shaft velocity [rpm]
x 10⁻³
Time [seconds]
Method: Radial Basis Function
Topology: [I8, H18, O6]
[radbas,purelin]
Stopcriteria: 0.20 (0.20)
Training: [17 epochs]
Date: 3–May–96
Train: testc_20.mat
Test: testc_21.mat
Type: Plot [r2c] (prediction)
Comment: –

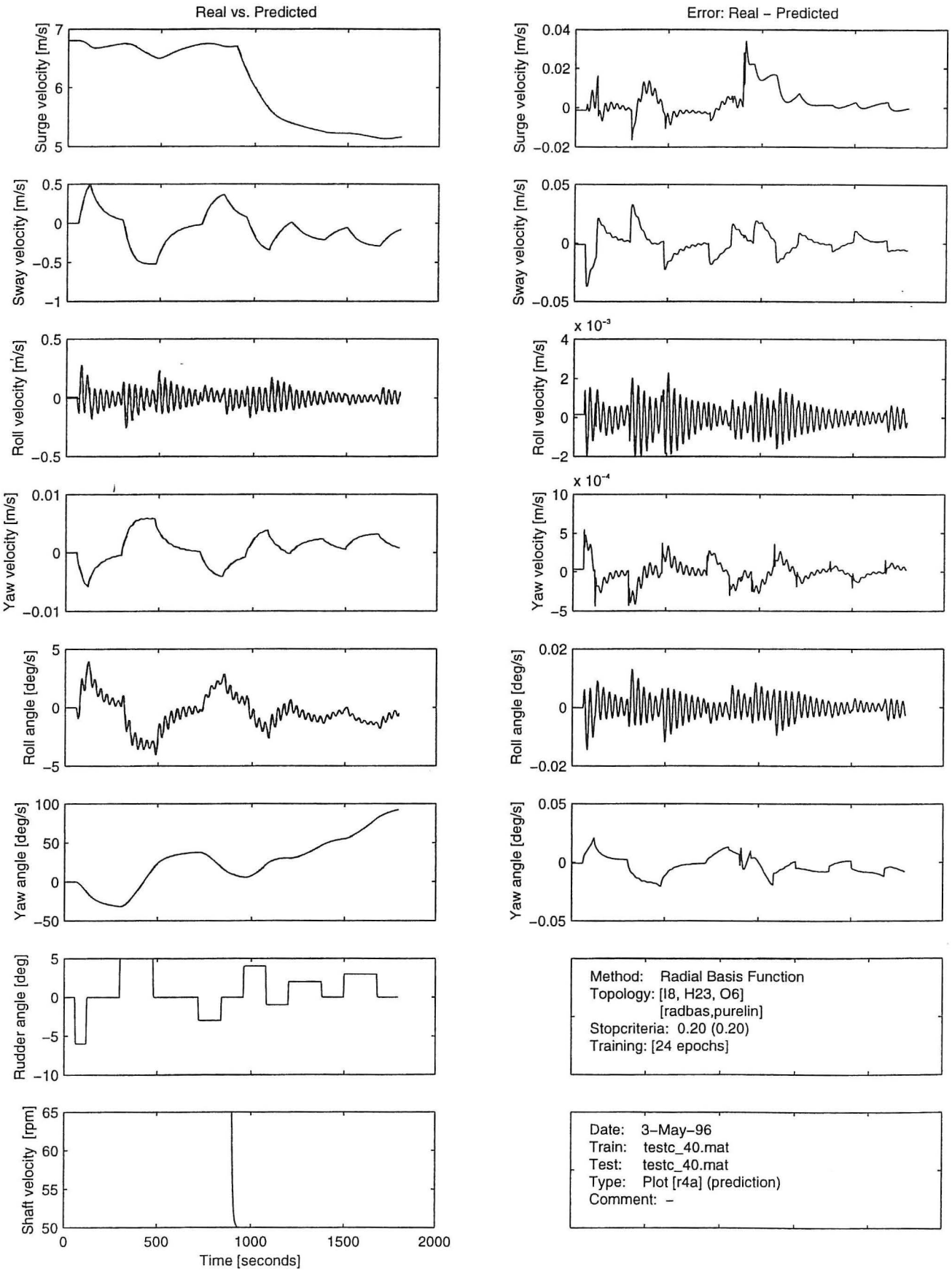

Real vs. Simulated
Error: Real – Simulated
Surge velocity [m/s]
Sway velocity [m/s]
Roll velocity [m/s]
Yaw velocity [m/s]
Roll angle [deg/s]
Yaw angle [deg/s]
Rudder angle [deg]
Shaft velocity [rpm]
x 10⁻³
Time [seconds]
Method: Radial Basis Function
Topology: [I8, H17, O6]
[radbas,purelin]
Stopcriteria: 0.20 (0.20)
Training: [18 epocs]
Date: 3-May-96
Train: testc_20.mat
Test: testc_21.mat
Type: Plot [r2d] (simulation)
Comment: –

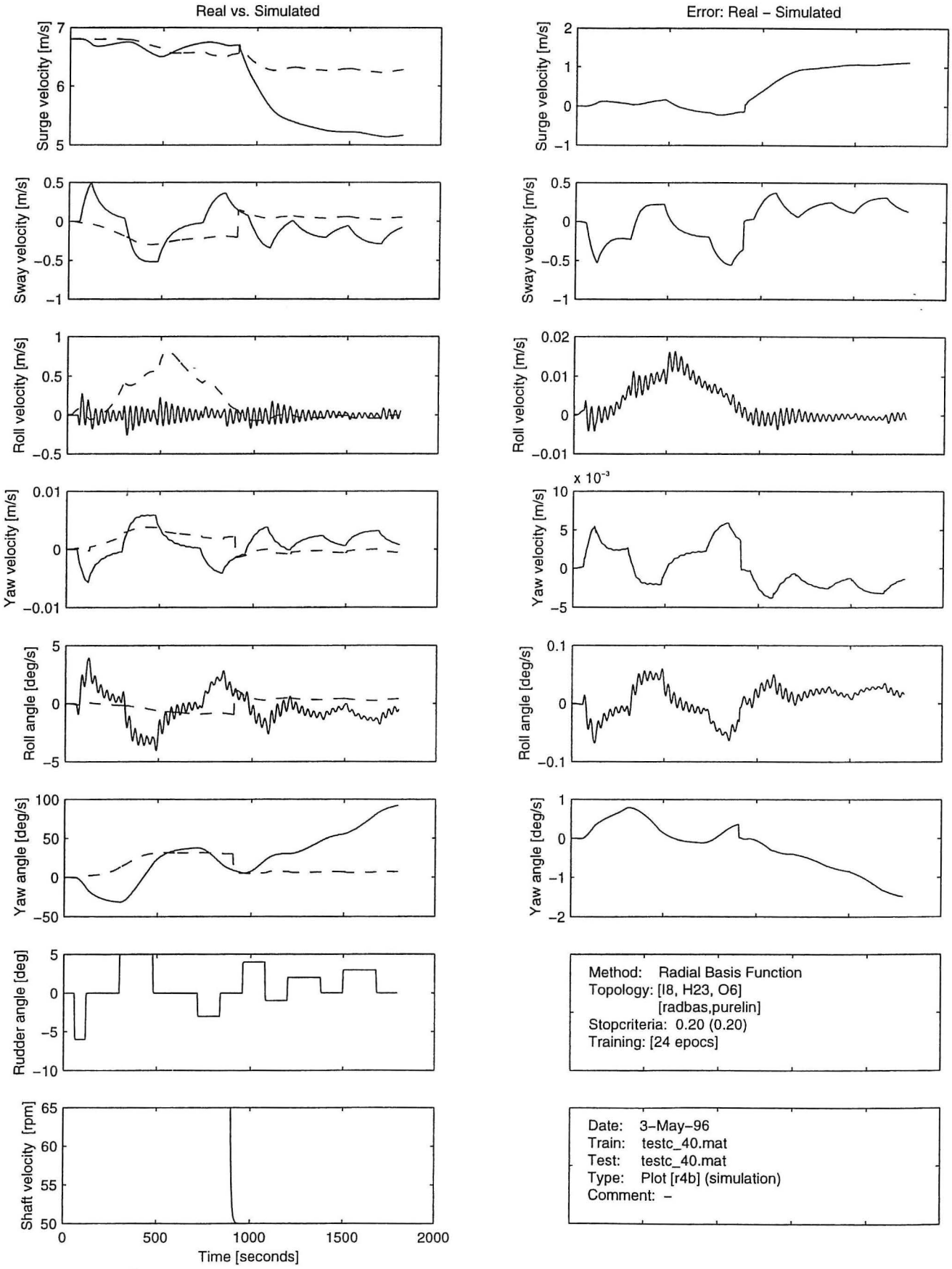

Real vs. Predicted
Error: Real – Predicted
Surge velocity [m/s]
Sway velocity [m/s]
Roll velocity [m/s]
Yaw velocity [m/s]
Roll angle [deg/s]
Yaw angle [deg/s]
Rudder angle [deg]
Shaft velocity [rpm]
x 10^-3
x 10^-4
Time [seconds]
Method: Radial Basis Function
Topology: [I8, H58, O6]
[radbas,purelin]
Stopcriteria: 0.20 (0.20)
Training: [59 epochs]
Date: 19–May–96
Train: testc_30.mat
Test: testc_30.mat
Type: Plot [r3a] (prediction)
Comment: –

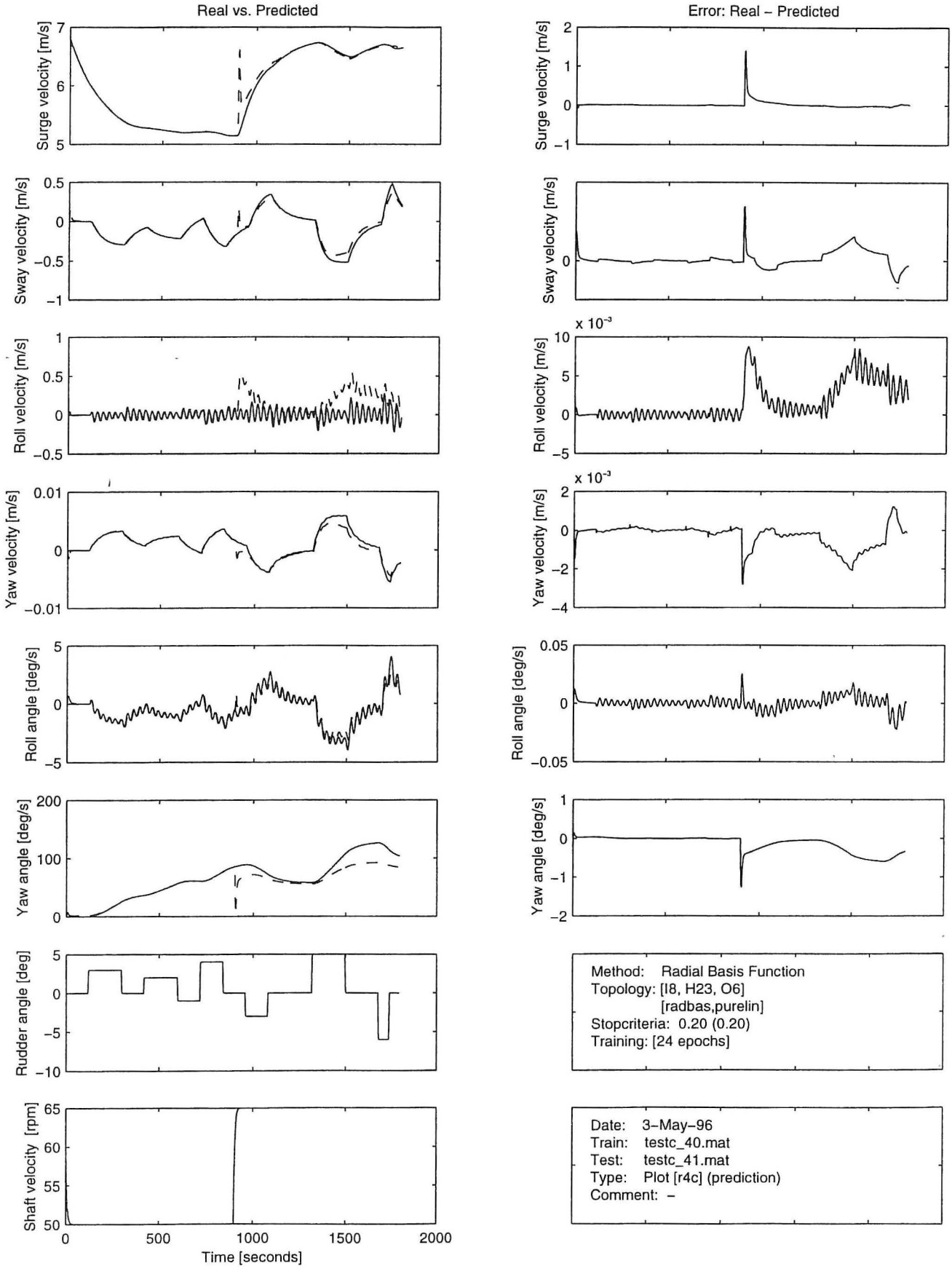

Real vs. Simulated
Error: Real – Simulated
Surge velocity [m/s]
Sway velocity [m/s]
Roll velocity [m/s]
x 10⁻³
Yaw velocity [m/s]
Roll angle [deg/s]
Yaw angle [deg/s]
Rudder angle [deg]
Shaft velocity [rpm]
Time [seconds]
Method: Radial Basis Function
Topology: [I8, H58, O6]
[radbas,purelin]
Stopcriteria: 0.20 (0.20)
Training: [59 epocs]
Date: 3-May-96
Train: testc_30.mat
Test: testc_30.mat
Type: Plot [r3b] (simulation)
Comment: –

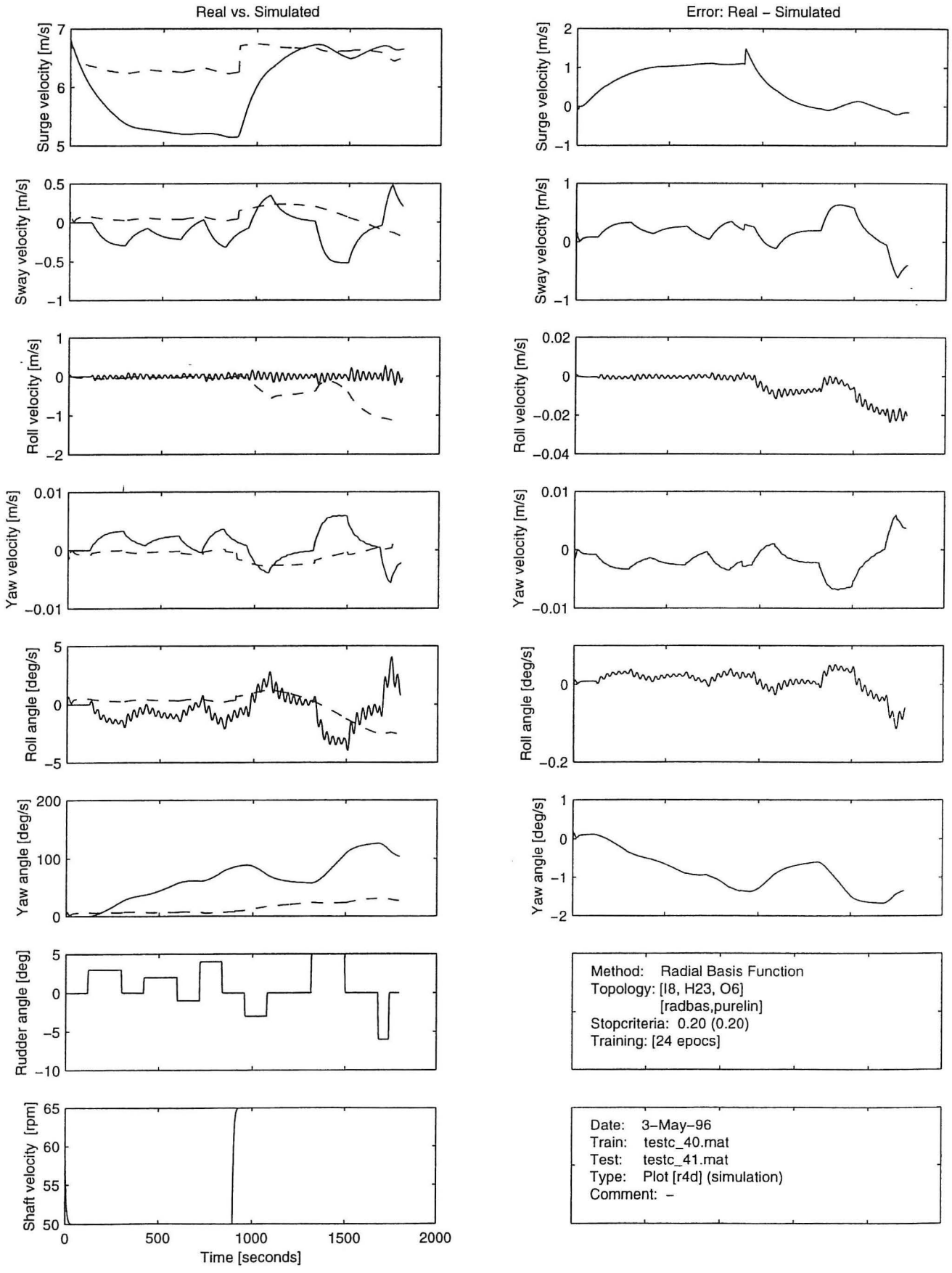

Real vs. Predicted
Error: Real – Predicted
Surge velocity [m/s]
Sway velocity [m/s]
Roll velocity [m/s]
Yaw velocity [m/s]
Roll angle [deg/s]
Yaw angle [deg/s]
Rudder angle [deg]
Shaft velocity [rpm]
x 10⁻³
x 10⁻⁴
Time [seconds]
Method: Radial Basis Function
Topology: [I8, H58, O6]
[radbas,purelin]
Stopcriteria: 0.20 (0.20)
Training: [59 epochs]
Date: 3–May–96
Train: testc_30.mat
Test: testc_31.mat
Type: Plot [r3c] (prediction)
Comment: –


Real vs. Simulated
Error: Real – Simulated
Surge velocity [m/s]
Sway velocity [m/s]
Roll velocity [m/s]
Yaw velocity [m/s]
x 10⁻³
Roll angle [deg/s]
Yaw angle [deg/s]
Rudder angle [deg]
Shaft velocity [rpm]
Time [seconds]
Method: Radial Basis Function
Topology: [I8, H58, O6]
[radbas,purelin]
Stopcriteria: 0.20 (0.20)
Training: [59 epocs]
Date: 3-May-96
Train: testc_30.mat
Test: testc_31.mat
Type: Plot [r3d] (simulation)
Comment: -

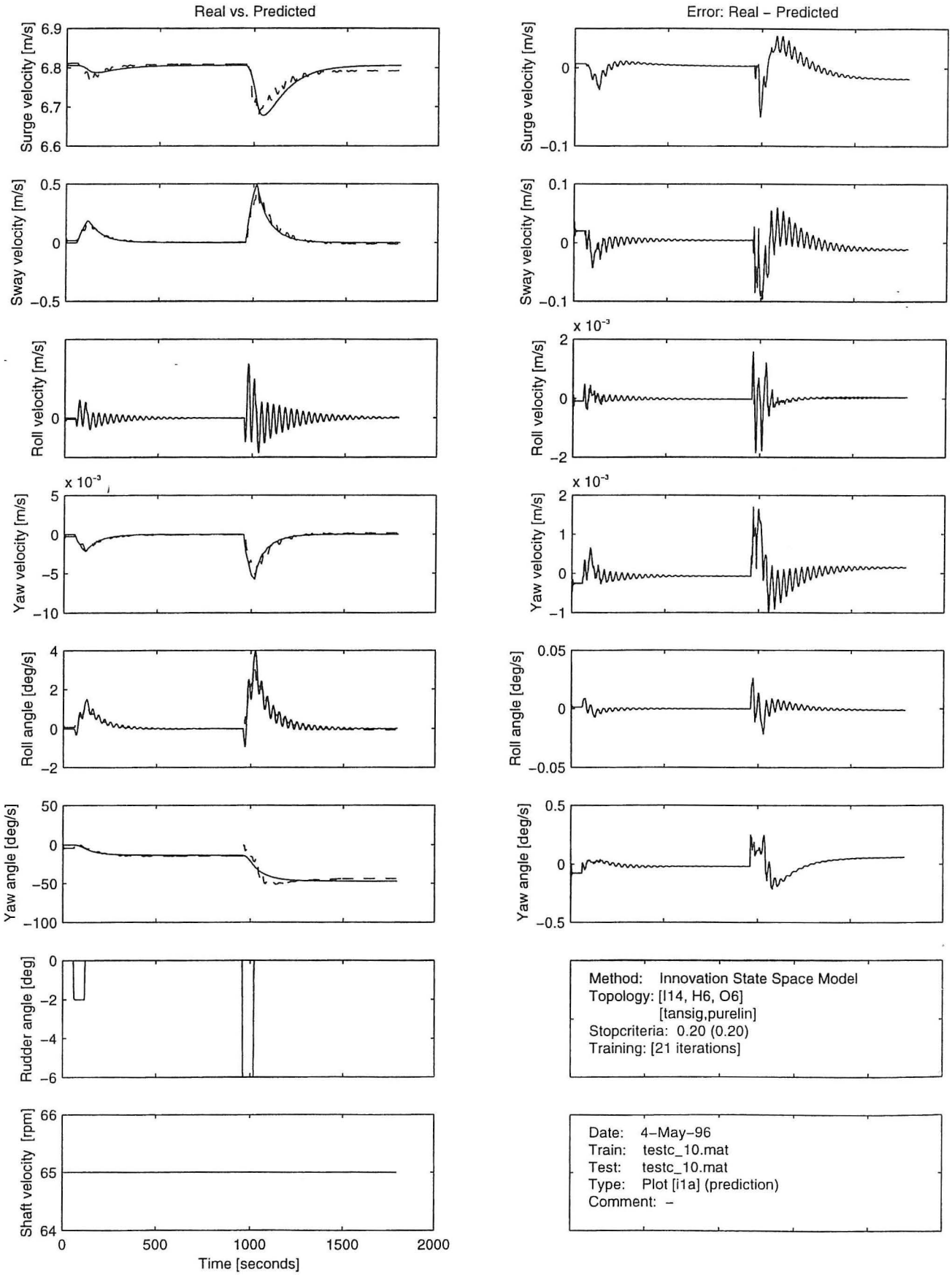

Real vs. Predicted
Error: Real – Predicted
Surge velocity [m/s]
Sway velocity [m/s]
Roll velocity [m/s]
Yaw velocity [m/s]
Roll angle [deg/s]
Yaw angle [deg/s]
Rudder angle [deg]
Shaft velocity [rpm]
x 10^-3
x 10^-4
Time [seconds]
Method: Radial Basis Function
Topology: [I8, H23, O6]
[radbas,purelin]
Stopcriteria: 0.20 (0.20)
Training: [24 epochs]
Date: 3-May-96
Train: testc_40.mat
Test: testc_40.mat
Type: Plot [r4a] (prediction)
Comment: –

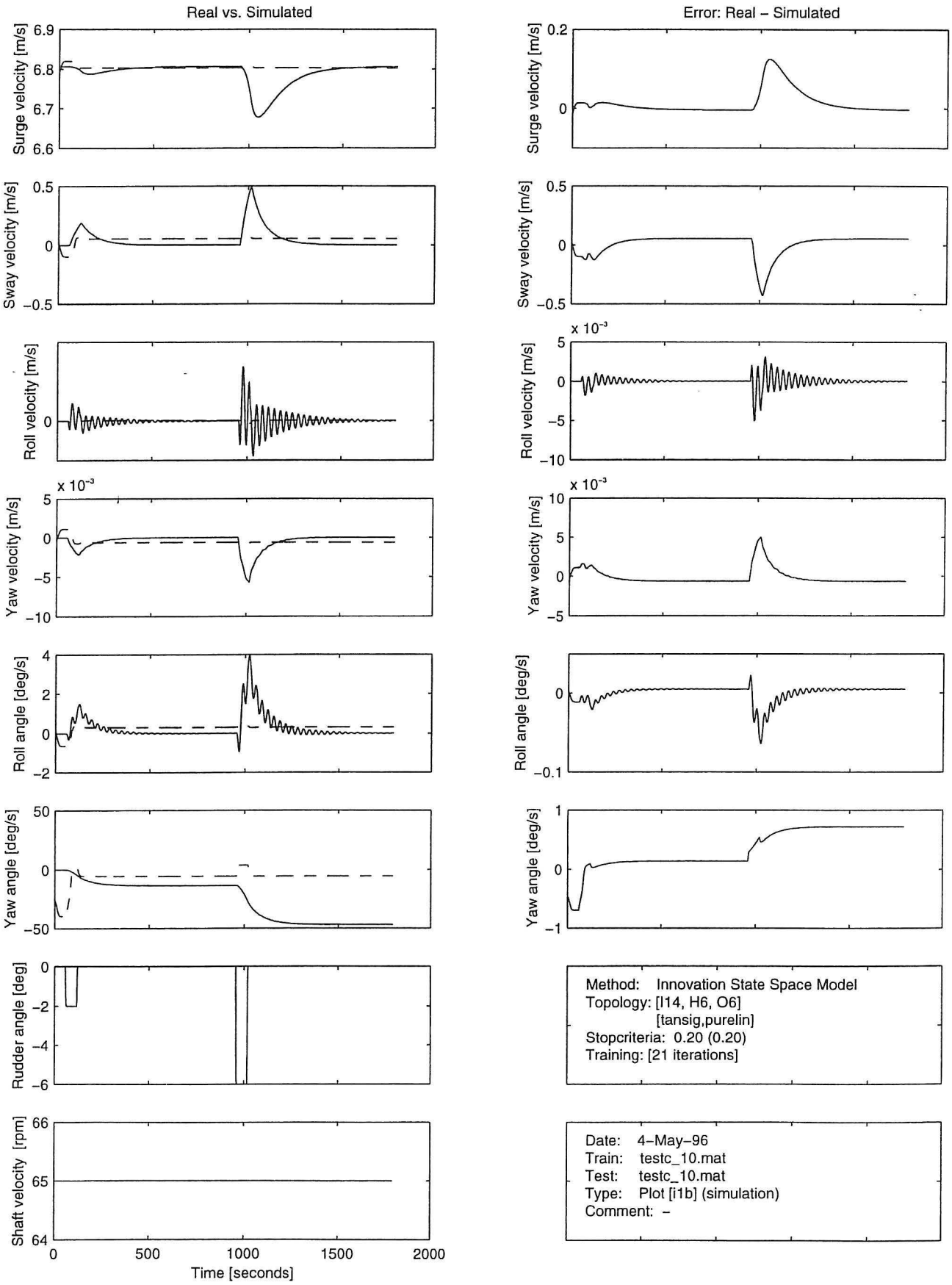

Real vs. Simulated
Error: Real – Simulated
Surge velocity [m/s]
Sway velocity [m/s]
Roll velocity [m/s]
Yaw velocity [m/s]
Roll angle [deg/s]
Yaw angle [deg/s]
Rudder angle [deg]
Shaft velocity [rpm]
x 10⁻³
Time [seconds]
Method: Radial Basis Function
Topology: [I8, H23, O6]
[radbas,purelin]
Stopcriteria: 0.20 (0.20)
Training: [24 epocs]
Date: 3-May-96
Train: testc_40.mat
Test: testc_40.mat
Type: Plot [r4b] (simulation)
Comment: –

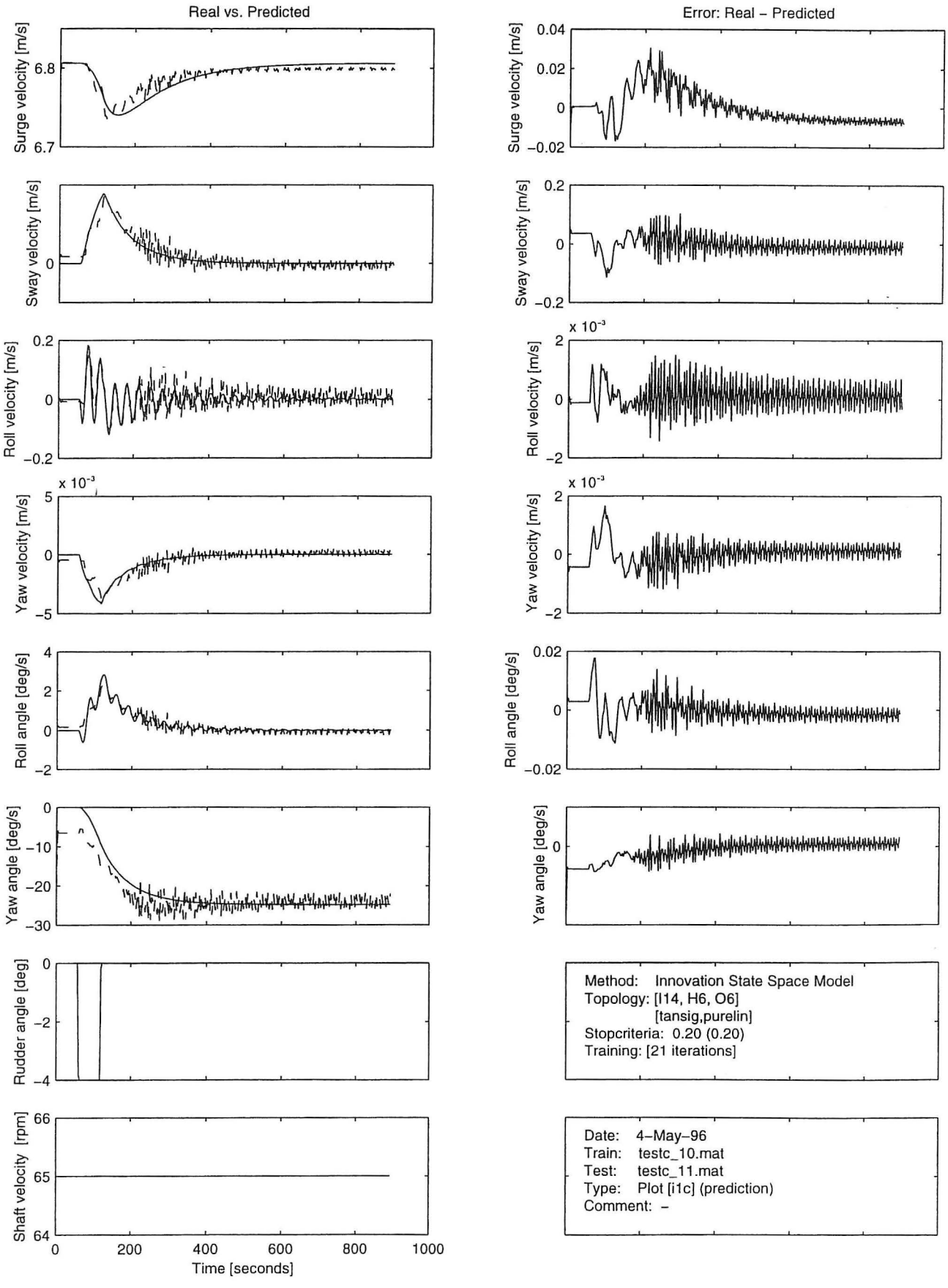

Real vs. Predicted
Error: Real – Predicted
Surge velocity [m/s]
Sway velocity [m/s]
Roll velocity [m/s]
Yaw velocity [m/s]
Roll angle [deg/s]
Yaw angle [deg/s]
Rudder angle [deg]
Shaft velocity [rpm]
x 10⁻³
Time [seconds]
Method: Radial Basis Function
Topology: [I8, H23, O6]
[radbas,purelin]
Stopcriteria: 0.20 (0.20)
Training: [24 epochs]
Date: 3-May-96
Train: testc_40.mat
Test: testc_41.mat
Type: Plot [r4c] (prediction)
Comment: –

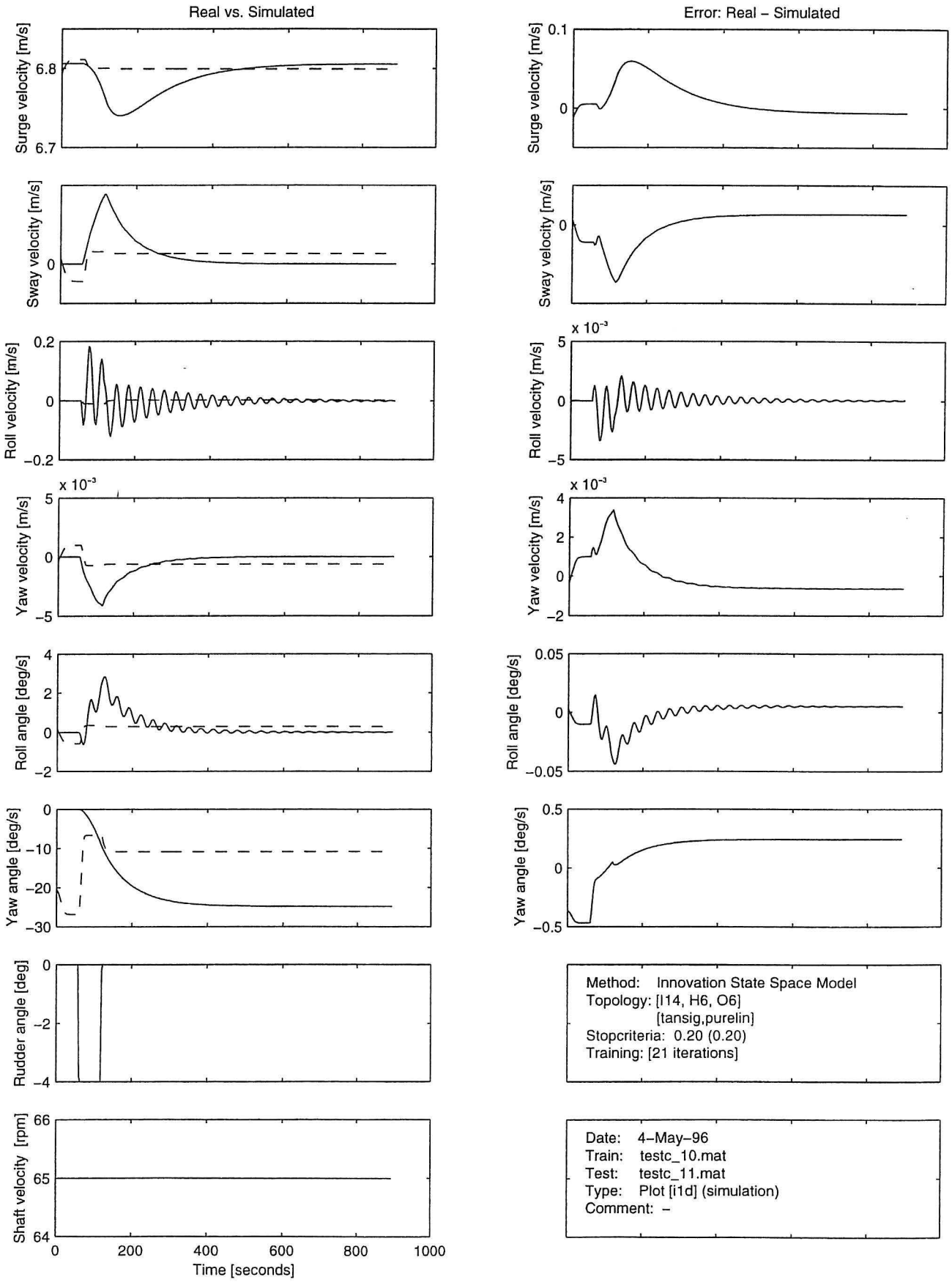

Real vs. Simulated
Error: Real – Simulated
Surge velocity [m/s]
Sway velocity [m/s]
Roll velocity [m/s]
Yaw velocity [m/s]
Roll angle [deg/s]
Yaw angle [deg/s]
Rudder angle [deg]
Shaft velocity [rpm]
Time [seconds]
Method: Radial Basis Function
Topology: [I8, H23, O6]
[radbas,purelin]
Stopcriteria: 0.20 (0.20)
Training: [24 epocs]
Date: 3-May-96
Train: testc_40.mat
Test: testc_41.mat
Type: Plot [r4d] (simulation)
Comment: –

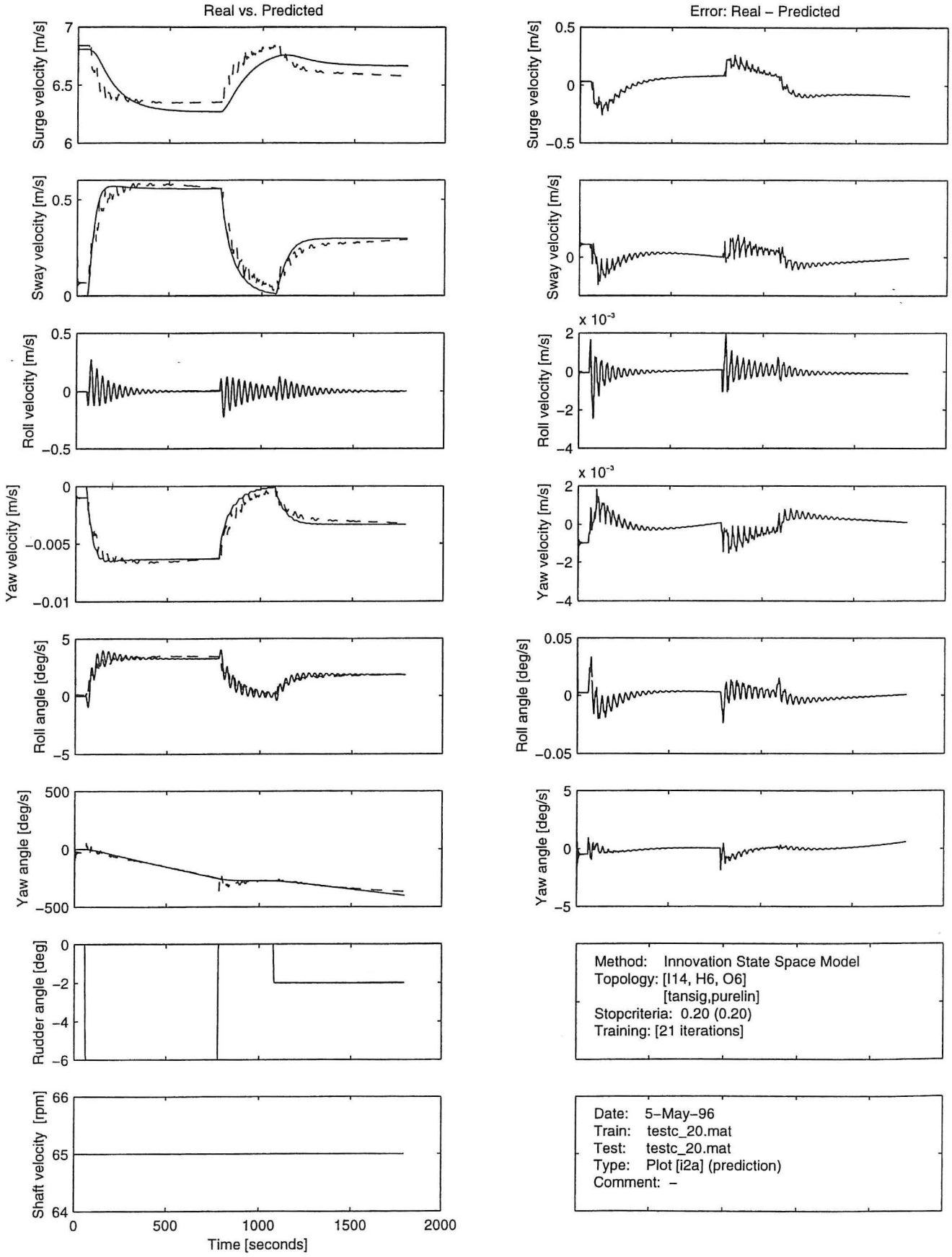
Appendix A.3 - RN Results

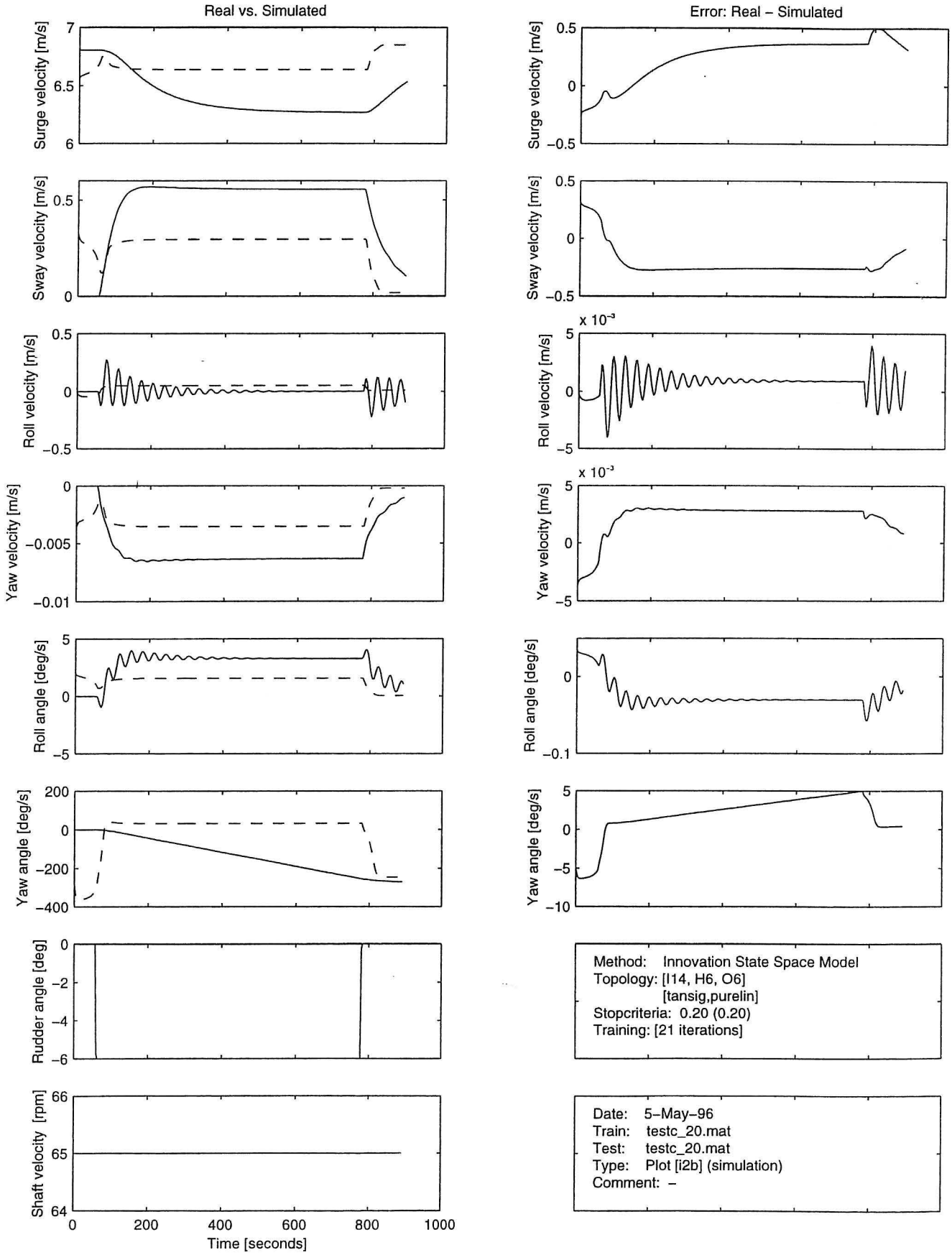


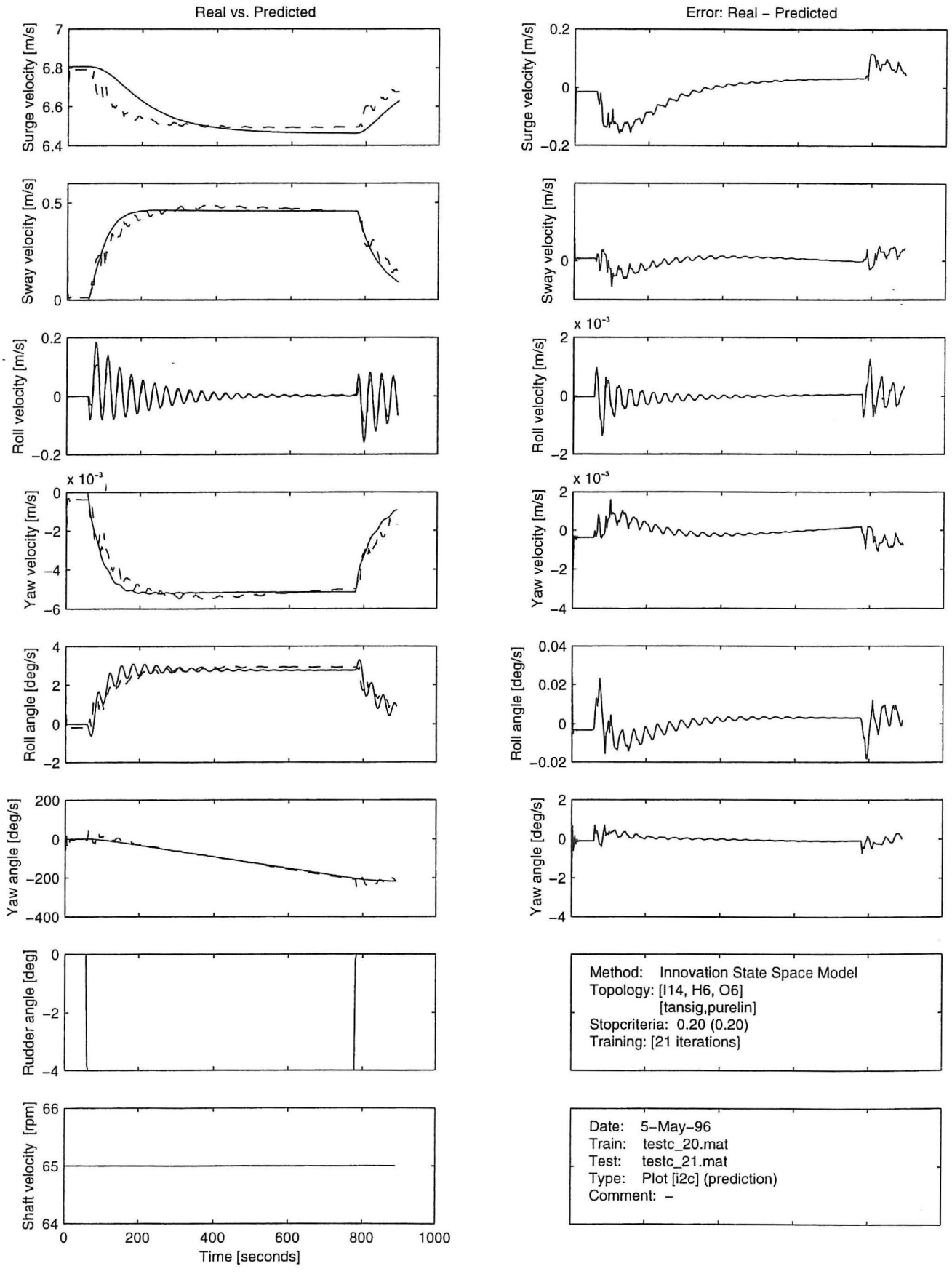

Real vs. Simulated
Error: Real – Simulated
Surge velocity [m/s]
Sway velocity [m/s]
Roll velocity [m/s]
Yaw velocity [m/s]
Roll angle [deg/s]
Yaw angle [deg/s]
Rudder angle [deg]
Shaft velocity [rpm]
x 10⁻³
Time [seconds]
Method: Innovation State Space Model
Topology: [I14, H6, O6]
[tansig,purelin]
Stopcriteria: 0.20 (0.20)
Training: [21 iterations]
Date: 4–May–96
Train: testc_10.mat
Test: testc_10.mat
Type: Plot [i1b] (simulation)
Comment: –

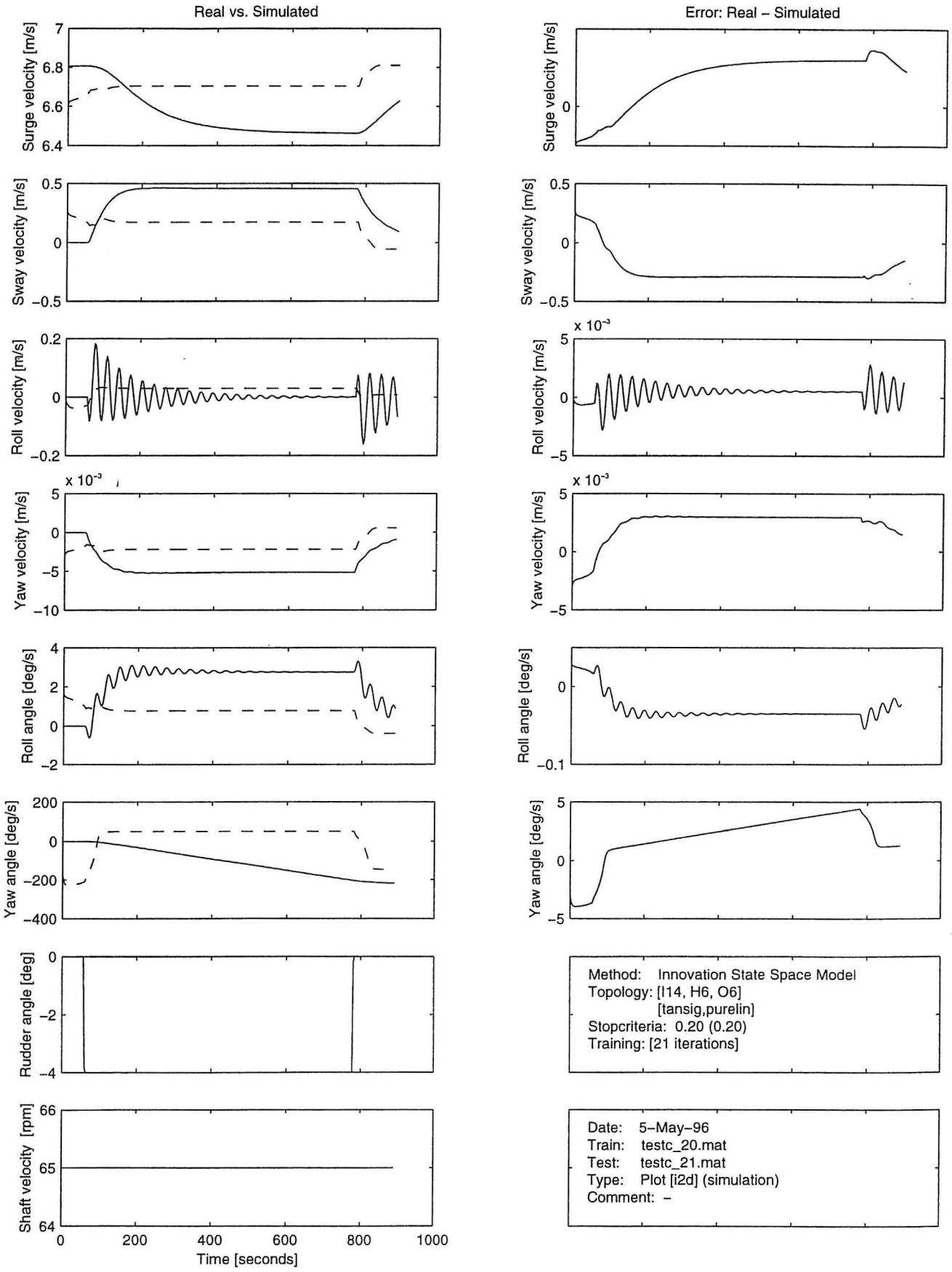

Real vs. Predicted
Error: Real – Predicted
Surge velocity [m/s]
Sway velocity [m/s]
Roll velocity [m/s]
Yaw velocity [m/s]
Roll angle [deg/s]
Yaw angle [deg/s]
Rudder angle [deg]
Shaft velocity [rpm]
x 10^-3
Time [seconds]
Method: Innovation State Space Model
Topology: [I14, H6, O6]
[tansig,purelin]
Stopcriteria: 0.20 (0.20)
Training: [21 iterations]
Date: 4–May–96
Train: testc_10.mat
Test: testc_11.mat
Type: Plot [i1c] (prediction)
Comment: –

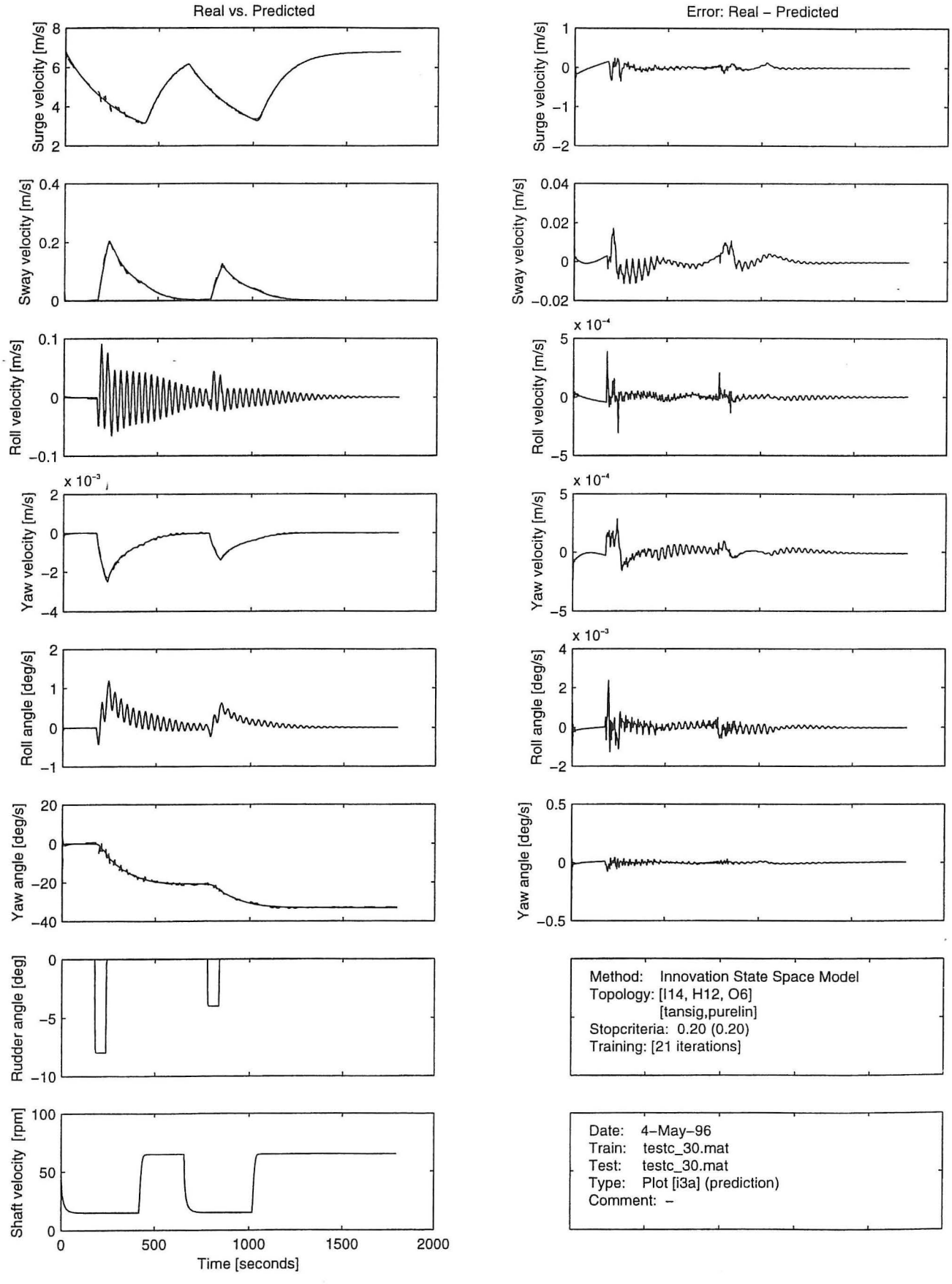

Real vs. Simulated
Error: Real – Simulated
Surge velocity [m/s]
6.8
6.7
0.1
0
Sway velocity [m/s]
0
Roll velocity [m/s]
0.2
0
−0.2
x 10⁻³
5
0
−5
Yaw velocity [m/s]
x 10⁻³
5
0
−5
4
2
0
−2
Roll angle [deg/s]
0.05
0
−0.05
Yaw angle [deg/s]
0
−10
−20
−30
0.5
0
−0.5
Rudder angle [deg]
0
−2
−4
Shaft velocity [rpm]
66
65
64
0
200
400
600
800
1000
Time [seconds]
Method: Innovation State Space Model
Topology: [I14, H6, O6]
[tansig,purelin]
Stopcriteria: 0.20 (0.20)
Training: [21 iterations]
Date: 4–May–96
Train: testc_10.mat
Test: testc_11.mat
Type: Plot [i1d] (simulation)
Comment: –

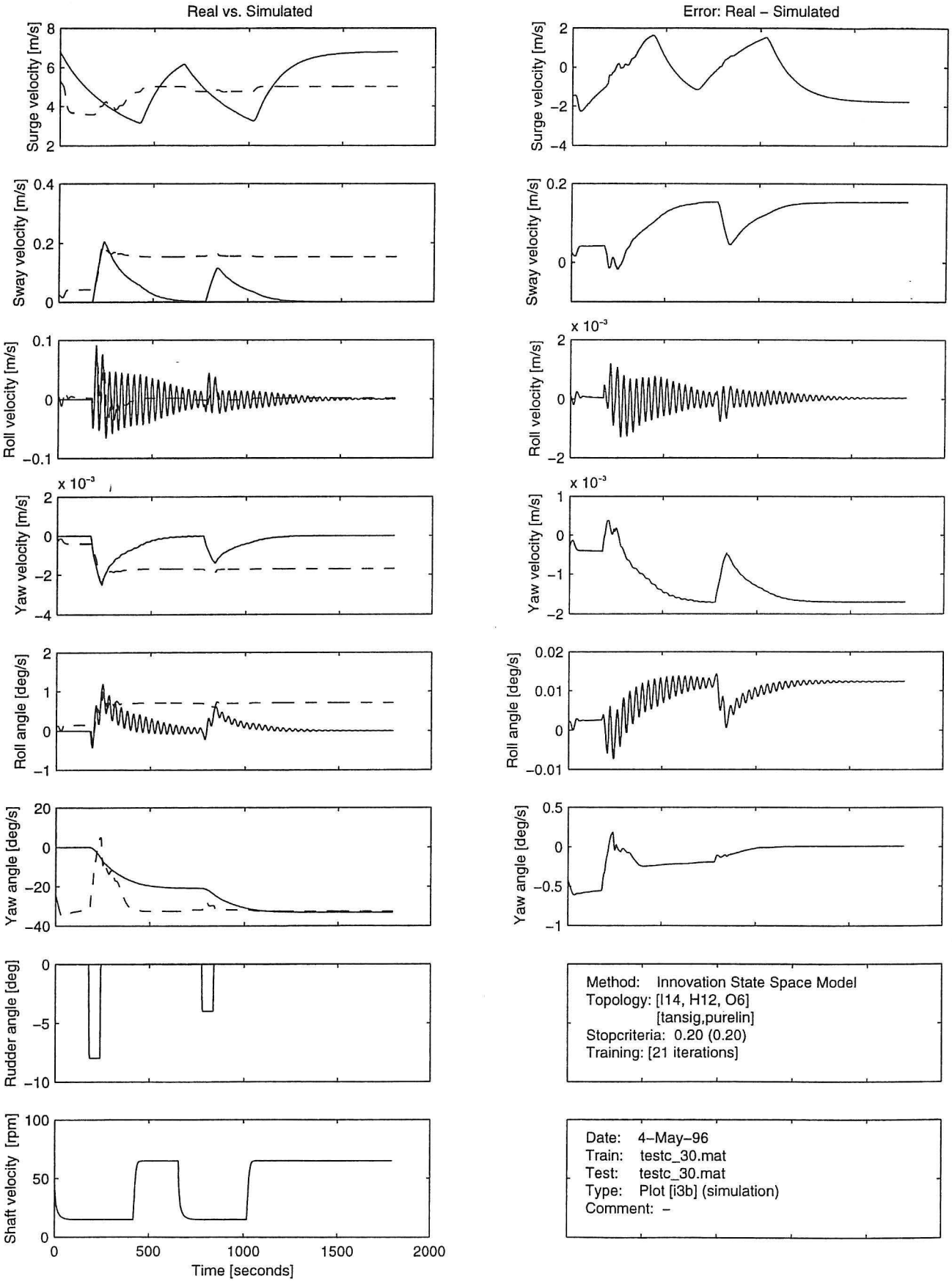

Real vs. Predicted
Error: Real – Predicted
Surge velocity [m/s]
Sway velocity [m/s]
Roll velocity [m/s]
Yaw velocity [m/s]
Roll angle [deg/s]
Yaw angle [deg/s]
Rudder angle [deg]
Shaft velocity [rpm]
x 10⁻³
Time [seconds]
Method: Innovation State Space Model
Topology: [I14, H6, O6]
[tansig,purelin]
Stopcriteria: 0.20 (0.20)
Training: [21 iterations]
Date: 5–May–96
Train: testc_20.mat
Test: testc_20.mat
Type: Plot [i2a] (prediction)
Comment: –

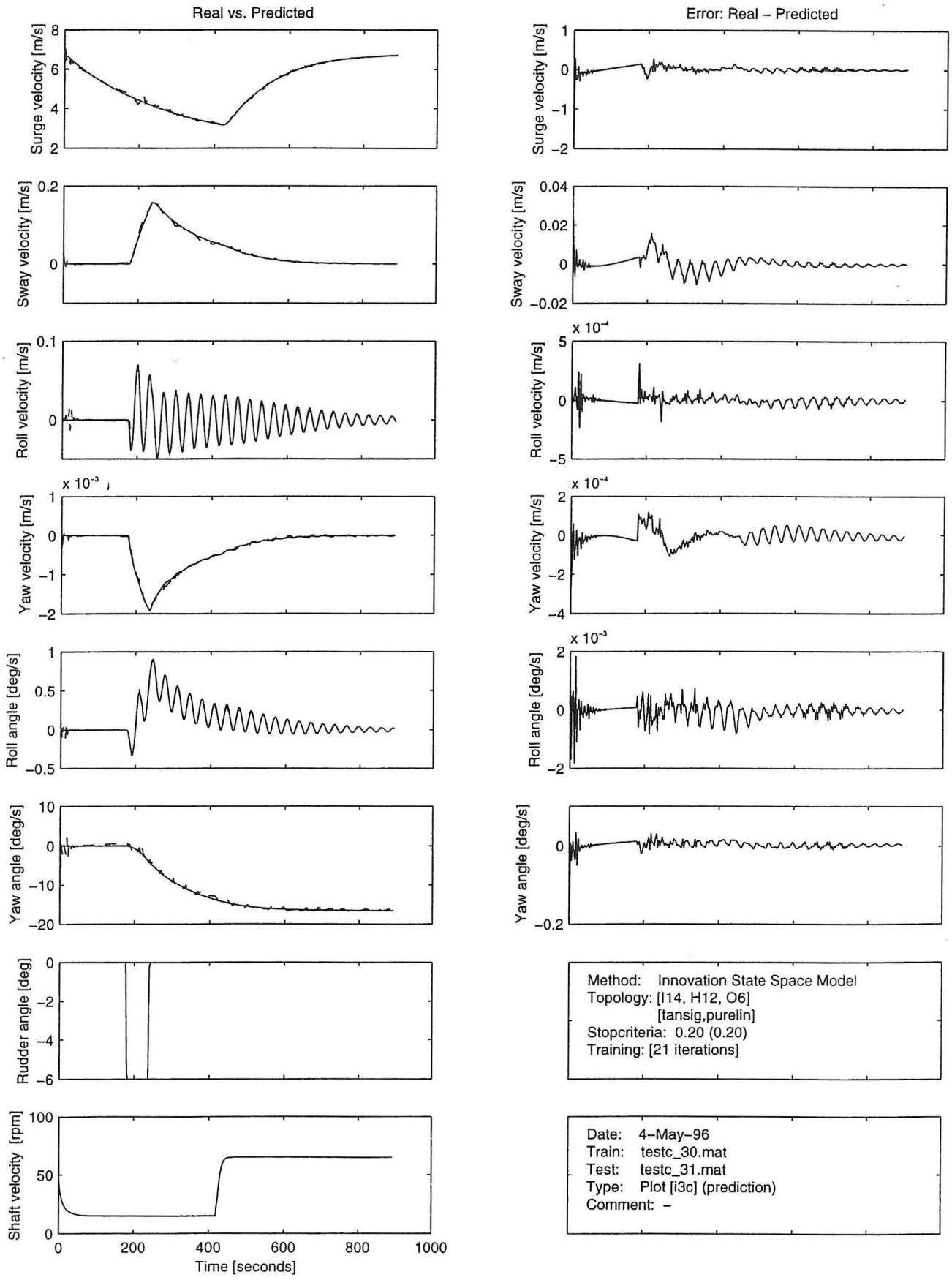

Real vs. Simulated
Error: Real – Simulated
Surge velocity [m/s]
Sway velocity [m/s]
Roll velocity [m/s]
Yaw velocity [m/s]
Roll angle [deg/s]
Yaw angle [deg/s]
Rudder angle [deg]
Shaft velocity [rpm]
x 10⁻³
Time [seconds]
Method: Innovation State Space Model
Topology: [I14, H6, O6]
[tansig,purelin]
Stopcriteria: 0.20 (0.20)
Training: [21 iterations]
Date: 5–May–96
Train: testc_20.mat
Test: testc_20.mat
Type: Plot [i2b] (simulation)
Comment: –

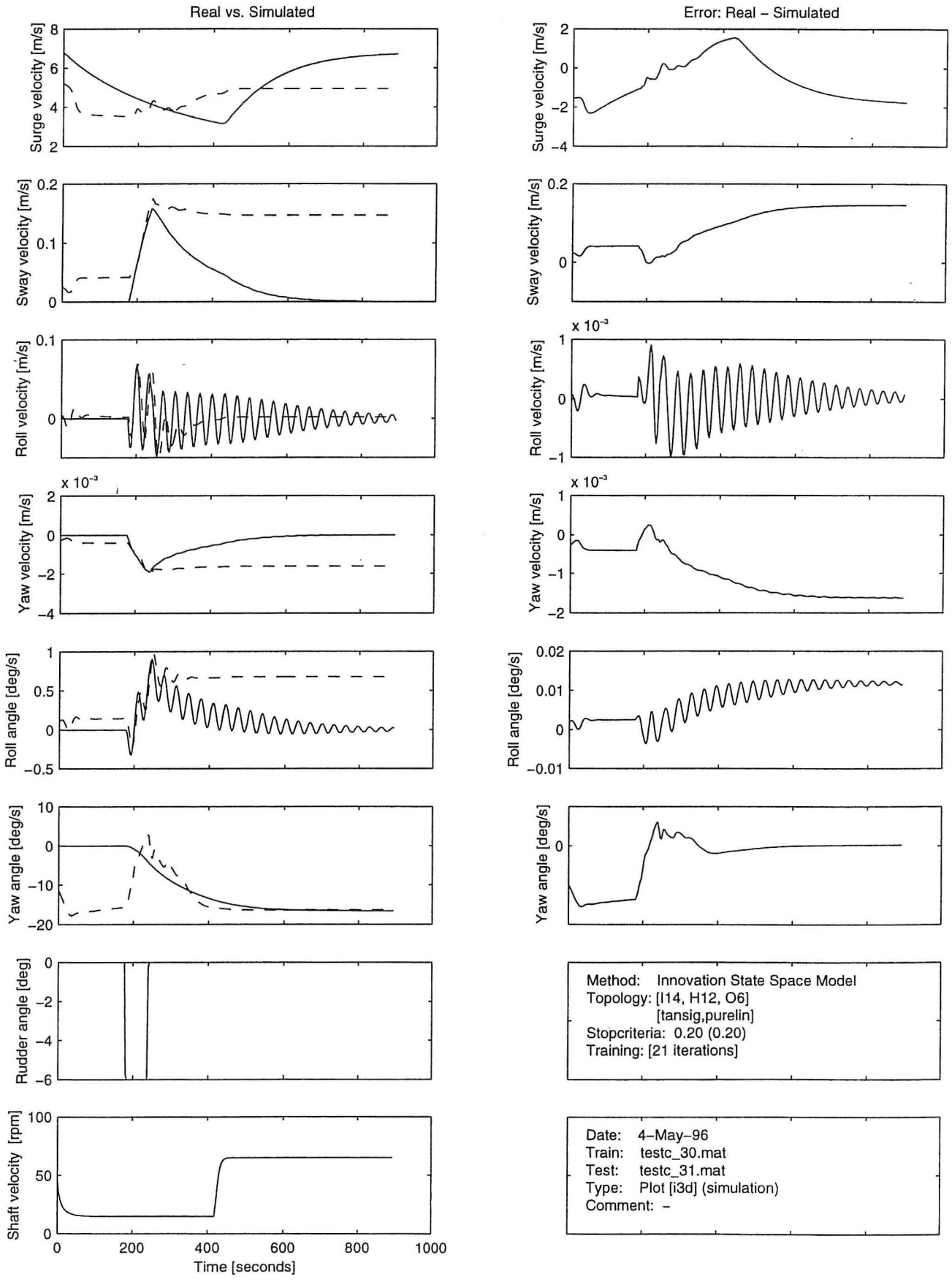

Real vs. Predicted
Error: Real – Predicted
Surge velocity [m/s]
Sway velocity [m/s]
Roll velocity [m/s]
Yaw velocity [m/s]
Roll angle [deg/s]
Yaw angle [deg/s]
Rudder angle [deg]
Shaft velocity [rpm]
x 10⁻³
Time [seconds]
Method: Innovation State Space Model
Topology: [I14, H6, O6]
[tansig,purelin]
Stopcriteria: 0.20 (0.20)
Training: [21 iterations]
Date: 5–May–96
Train: testc_20.mat
Test: testc_21.mat
Type: Plot [i2c] (prediction)
Comment: –

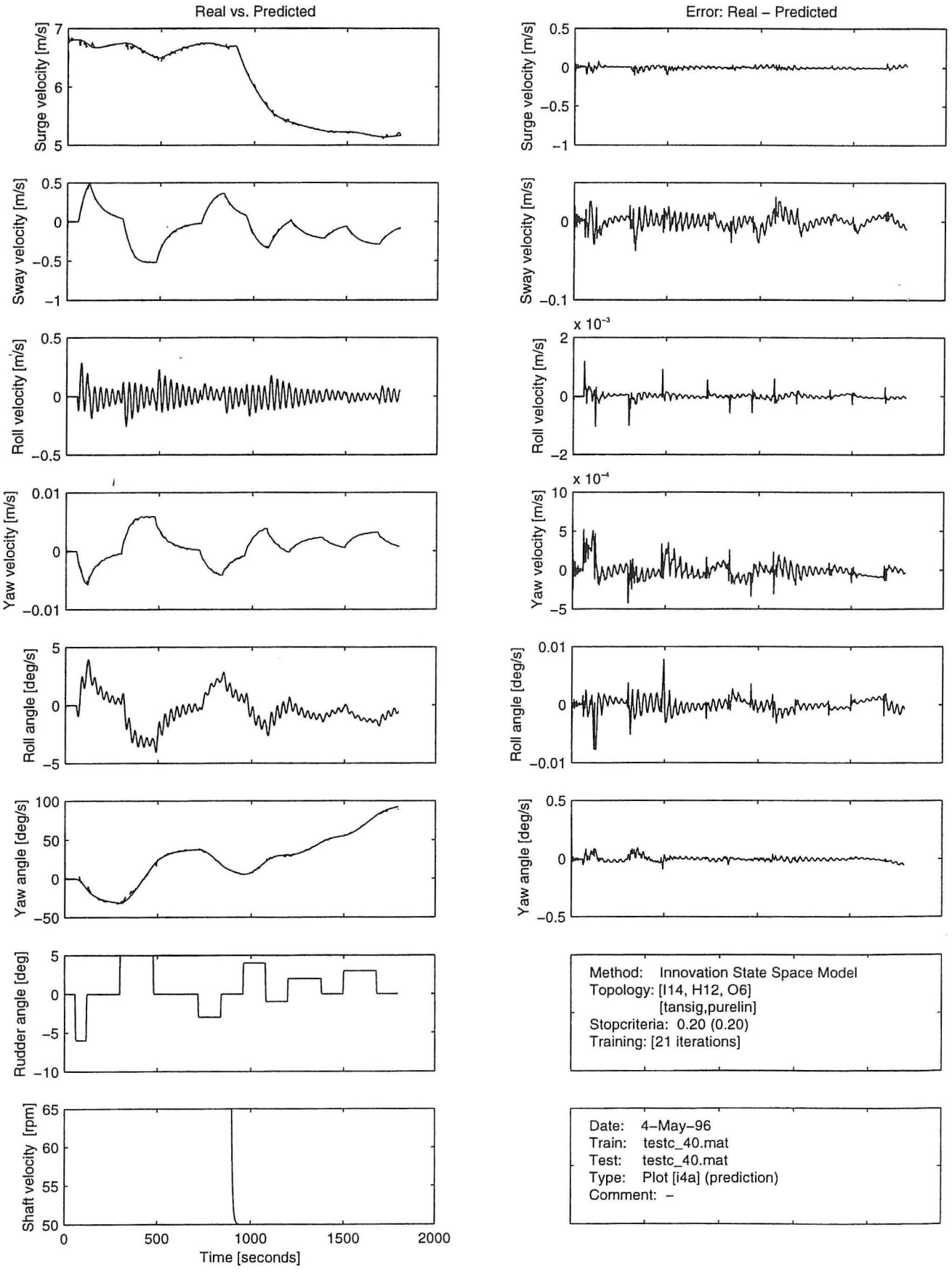

Real vs. Simulated
Error: Real – Simulated
Surge velocity [m/s]
Sway velocity [m/s]
Roll velocity [m/s]
Yaw velocity [m/s]
Roll angle [deg/s]
Yaw angle [deg/s]
Rudder angle [deg]
Shaft velocity [rpm]
x 10⁻³
Time [seconds]
Method: Innovation State Space Model
Topology: [I14, H6, O6]
[tansig,purelin]
Stopcriteria: 0.20 (0.20)
Training: [21 iterations]
Date: 5–May–96
Train: testc_20.mat
Test: testc_21.mat
Type: Plot [i2d] (simulation)
Comment: –

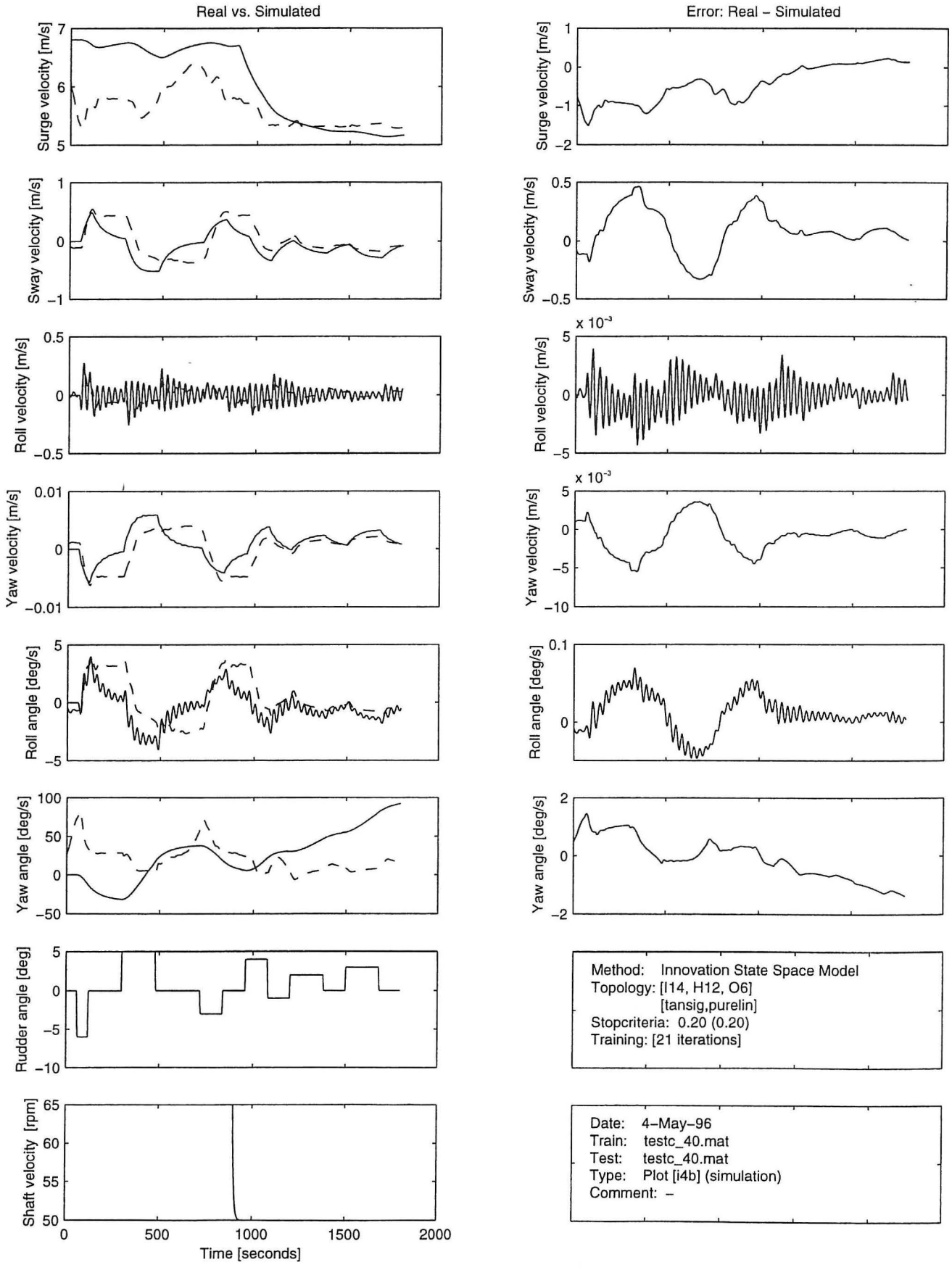

Real vs. Predicted
Error: Real – Predicted
Surge velocity [m/s]
Sway velocity [m/s]
Roll velocity [m/s]
Yaw velocity [m/s]
Roll angle [deg/s]
Yaw angle [deg/s]
Rudder angle [deg]
Shaft velocity [rpm]
x 10⁻³
x 10⁻⁴
Time [seconds]
Method: Innovation State Space Model
Topology: [I14, H12, O6]
[tansig,purelin]
Stopcriteria: 0.20 (0.20)
Training: [21 iterations]
Date: 4-May-96
Train: testc_30.mat
Test: testc_30.mat
Type: Plot [i3a] (prediction)
Comment: –

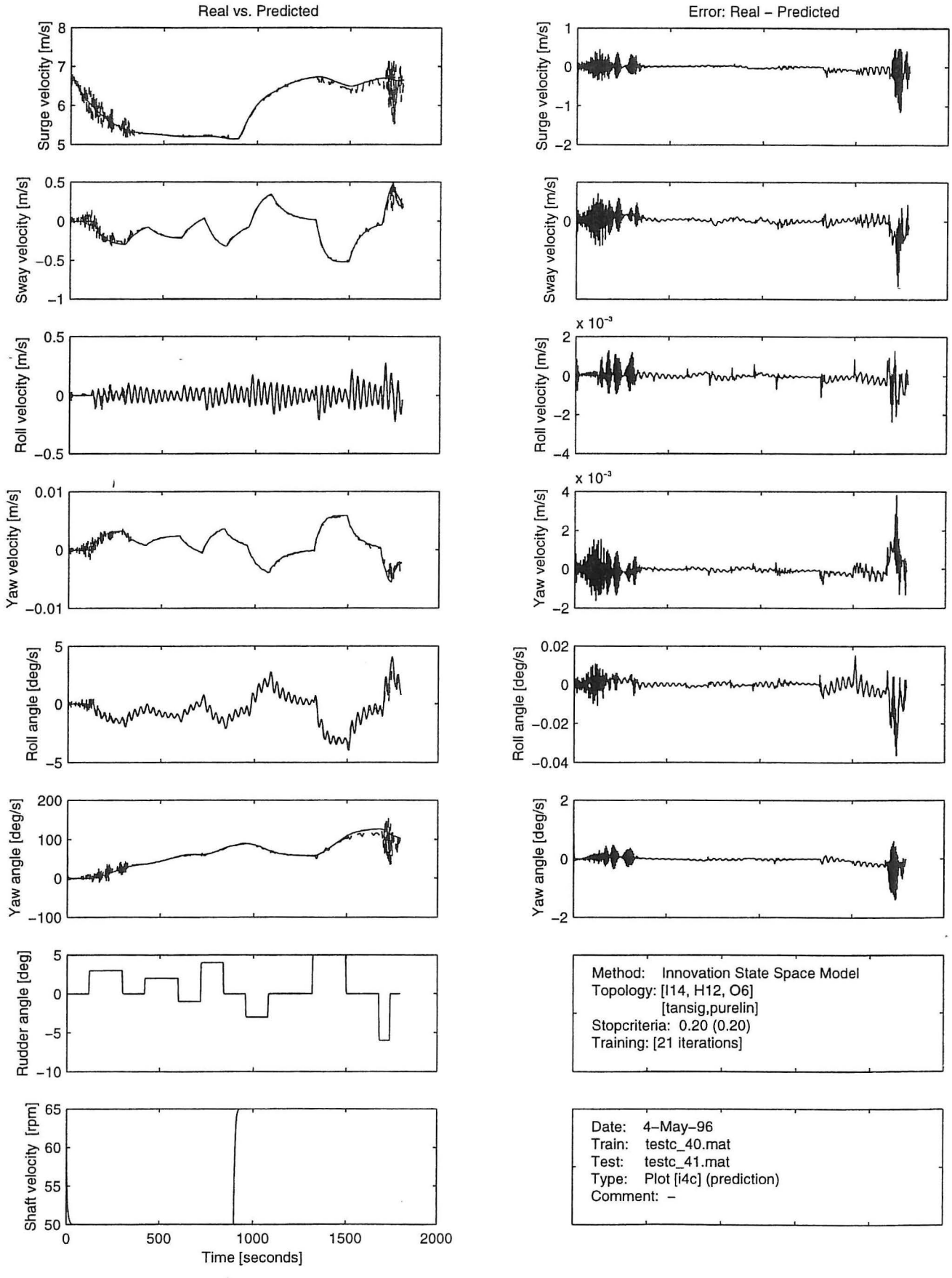

Real vs. Simulated
Error: Real – Simulated
Surge velocity [m/s]
Sway velocity [m/s]
Roll velocity [m/s]
Yaw velocity [m/s]
Roll angle [deg/s]
Yaw angle [deg/s]
Rudder angle [deg]
Shaft velocity [rpm]
x 10⁻³
Time [seconds]
Method: Innovation State Space Model
Topology: [I14, H12, O6]
[tansig,purelin]
Stopcriteria: 0.20 (0.20)
Training: [21 iterations]
Date: 4–May–96
Train: testc_30.mat
Test: testc_30.mat
Type: Plot [i3b] (simulation)
Comment: –

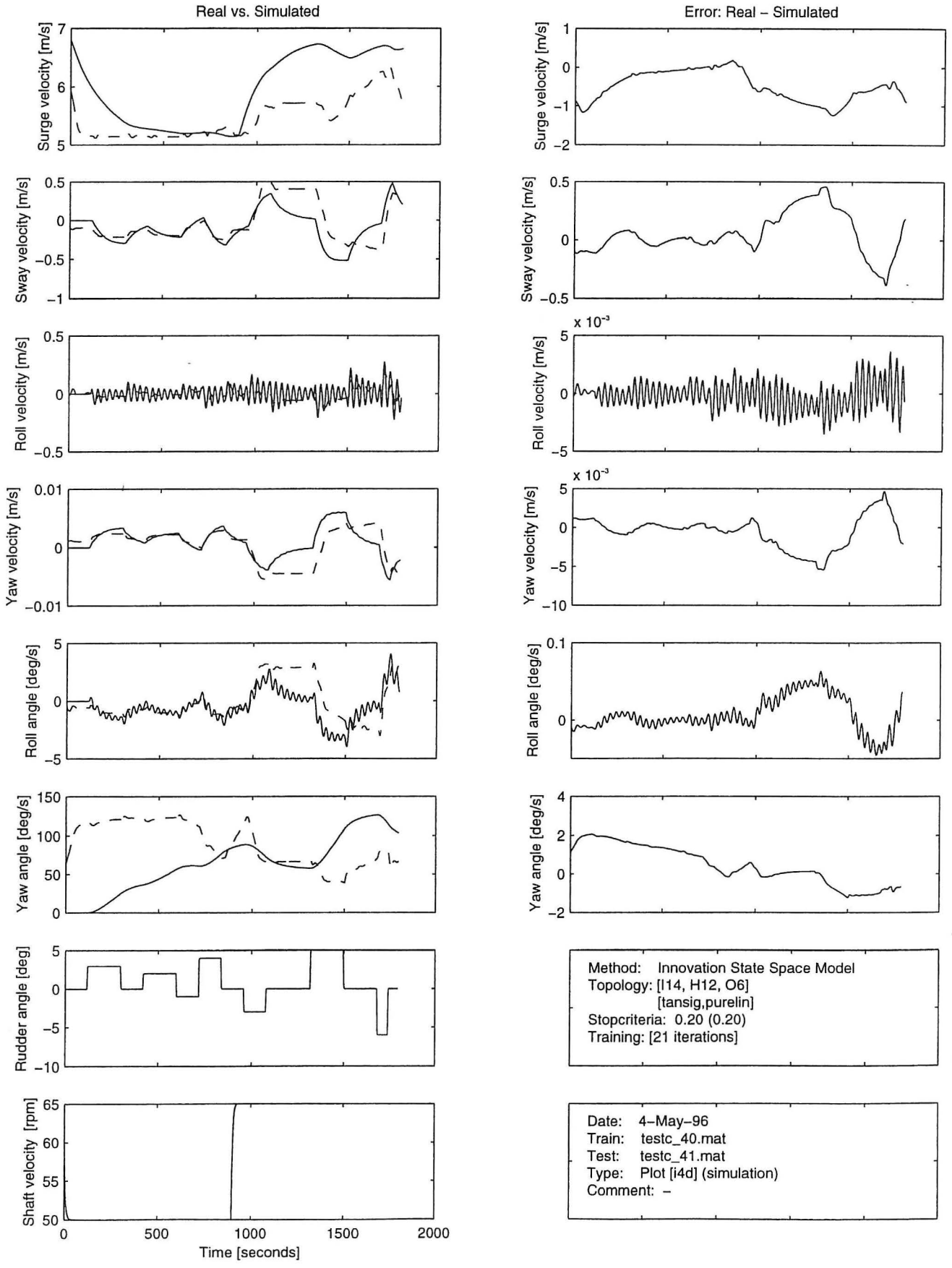

Real vs. Predicted
Error: Real – Predicted
Surge velocity [m/s]
Sway velocity [m/s]
Roll velocity [m/s]
Yaw velocity [m/s]
Roll angle [deg/s]
Yaw angle [deg/s]
Rudder angle [deg]
Shaft velocity [rpm]
x 10^-3
x 10^-4
Time [seconds]
Method: Innovation State Space Model
Topology: [I14, H12, O6]
[tansig,purelin]
Stopcriteria: 0.20 (0.20)
Training: [21 iterations]
Date: 4–May–96
Train: testc_30.mat
Test: testc_31.mat
Type: Plot [i3c] (prediction)
Comment: –


Real vs. Simulated
Error: Real – Simulated
Surge velocity [m/s]
Sway velocity [m/s]
Roll velocity [m/s]
Yaw velocity [m/s]
Roll angle [deg/s]
Yaw angle [deg/s]
Rudder angle [deg]
Shaft velocity [rpm]
x 10⁻³
Time [seconds]
Method: Innovation State Space Model
Topology: [I14, H12, O6]
[tansig,purelin]
Stopcriteria: 0.20 (0.20)
Training: [21 iterations]
Date: 4–May–96
Train: testc_30.mat
Test: testc_31.mat
Type: Plot [i3d] (simulation)
Comment: –


Real vs. Predicted
Error: Real – Predicted
Surge velocity [m/s]
Sway velocity [m/s]
Roll velocity [m/s]
Yaw velocity [m/s]
Roll angle [deg/s]
Yaw angle [deg/s]
Rudder angle [deg]
Shaft velocity [rpm]
x 10^-3
x 10^-4
Time [seconds]
Method: Innovation State Space Model
Topology: [I14, H12, O6]
[tansig,purelin]
Stopcriteria: 0.20 (0.20)
Training: [21 iterations]
Date: 4–May–96
Train: testc_40.mat
Test: testc_40.mat
Type: Plot [i4a] (prediction)
Comment: –


Real vs. Simulated
Error: Real – Simulated
Surge velocity [m/s]
Sway velocity [m/s]
Roll velocity [m/s]
Yaw velocity [m/s]
Roll angle [deg/s]
Yaw angle [deg/s]
Rudder angle [deg]
Shaft velocity [rpm]
x 10⁻³
Time [seconds]
Method: Innovation State Space Model
Topology: [I14, H12, O6]
[tansig,purelin]
Stopcriteria: 0.20 (0.20)
Training: [21 iterations]
Date: 4–May–96
Train: testc_40.mat
Test: testc_40.mat
Type: Plot [i4b] (simulation)
Comment: –


Real vs. Predicted
Error: Real – Predicted
Surge velocity [m/s]
Sway velocity [m/s]
Roll velocity [m/s]
Yaw velocity [m/s]
Roll angle [deg/s]
Yaw angle [deg/s]
Rudder angle [deg]
Shaft velocity [rpm]
x 10⁻³
Time [seconds]
Method: Innovation State Space Model
Topology: [I14, H12, O6]
[tansig,purelin]
Stopcriteria: 0.20 (0.20)
Training: [21 iterations]
Date: 4–May–96
Train: testc_40.mat
Test: testc_41.mat
Type: Plot [i4c] (prediction)
Comment: –


Real vs. Simulated
Error: Real – Simulated
Surge velocity [m/s]
Sway velocity [m/s]
Roll velocity [m/s]
Yaw velocity [m/s]
Roll angle [deg/s]
Yaw angle [deg/s]
Rudder angle [deg]
Shaft velocity [rpm]
x 10⁻³
Time [seconds]
Method: Innovation State Space Model
Topology: [I14, H12, O6]
[tansig,purelin]
Stopcriteria: 0.20 (0.20)
Training: [21 iterations]
Date: 4–May–96
Train: testc_40.mat
Test: testc_41.mat
Type: Plot [i4d] (simulation)
Comment: –

STRUCTURAL RELIABILITY THEORY SERIES

PAPER NO. 127: H. U. Köylüoğlu, S. R. K. Nielsen and A. Ş. Çakmak: *A Faster Simulation Method for the Stochastic Response of Hysteretic Structures subject to Earthquakes.* ISSN 0902-7513 R9523.

PAPER NO. 128: H. U. Köylüoğlu, S. R. K. Nielsen, A. Ş. Çakmak & P. H. Kirkegaard: *Prediction of Global and Localized Damage and Future Reliability for RC Structures subject to Earthquakes.* ISSN 0901-7513 R9426.

PAPER NO. 129: C. Pedersen & P. Thoft-Christensen: *Interactive Structural Optimization with Quasi-Newton Algorithms.* ISSN 0902-7513 R9436.

PAPER NO. 130: I. Enevoldsen & J. D. Sørensen: *Decomposition Techniques and Effective Algorithms in Reliability-Based Optimization.* ISSN 0902-7513 R9412.

PAPER NO. 131: H. U. Köylüoğlu, S. R. K. Nielsen & A. Ş. Çakmak: *Approximate Forward Difference Equations for the Lower Order Non-Stationary Statistics of Geometrically Non-Linear Systems subject to Random Excitation.* ISSN 0902-7513 R9422.

PAPER NO. 132: I. B. Kroon: *Decision Theory applied to Structural Engineering Problems.* Ph.D.-Thesis. ISSN 0902-7513 R9421.

PAPER 133: H. U. Köylüoğlu, S. R. K. Nielsen & A. Ş. Çakmak: *Stochastic Dynamics of Nonlinear Structures with Random Properties subject to Random Stationary Excitation.* ISSN 0902-7513 R9520.

PAPER NO. 134: H. U. Köylüoğlu, S. R. K. Nielsen & A. Ş. Çakmak: *Solution of Random Structural System subject to Non-Stationary Excitation: Transforming the Equation with Random Coefficients to One with Deterministic Coefficients and Random Initial Conditions.* ISSN 0902-7513 R9429.

PAPER NO. 135: S. Engelund, J. D. Sørensen & S. Krenk: *Estimation of the Time to Initiation of Corrosion in Existing Uncracked Concrete Structures.* ISSN 0902-7513 R9438.

PAPER NO. 136: H. U. Köylüoğlu, S. R. K. Nielsen & A. Ş. Çakmak:
Solution Methods for Structures with Random Properties subject to Random Excitation. ISSN 0902-7513 R9444.

PAPER NO. 137: J. D. Sørensen, M. H. Faber & I. B. Kroon:
Optimal Reliability-Based Planning of Experiments for POD Curves. ISSN 0902-7513 R9455.

PAPER NO. 138: S.R.K. Nielsen & P.S. Skjærbæk, H.U. Köylüoğlu & A.Ş. Çakmak: *Prediction of Global Damage and Reliability based upon Sequential Identification and Updating of RC Structures subject to Earthquakes.* ISSN 0902-7513 R9505.

PAPER NO. 139: R. Iwankiewicz, S. R. K. Nielsen & P. S. Skjærbæk: *Sensitivity of Reliability Estimates in Partially Damaged RC Structures subject to Earthquakes, using Reduced Hysteretic Models.* ISSN 0902-7513 R9507.

PAPER NO 141: H. U. Köylüoğlu, S. R. K. Nielsen & A. Ş. Çakmak: *Uncertain Buckling Load and Reliability of Columns with Uncertain Properties.* ISSN 0902-7513 R9524.

STRUCTURAL RELIABILITY THEORY SERIES

PAPER NO. 142: S. R. K. Nielsen & R. Iwankiewicz: *Response of Non-Linear Systems to Renewal Impulses by Path Integration.* ISSN 0902-7513 R9512.

PAPER NO. 145: H. U. Köylüoğlu, S. R. K. Nielsen, Jamison Abbott and A. Ş. Çakmak: *Local and Modal Damage Indicators for Reinforced Concrete Shear Frames subject to Earthquakes.* ISSN 0902-7513 R9521

PAPER NO. 146: P. H. Kirkegaard, S. R. K. Nielsen, R. C. Micaletti and A. Ş. Çakmak: *Identification of a Maximum Softening Damage Indicator of RC-Structures using Time-Frequency Techniques.* ISSN 0902-7513 R9522.

PAPER NO. 147: R. C. Micaletti, A. Ş. Çakmak, S. R. K. Nielsen & P. H. Kirkegaard: *Construction of Time-Dependent Spectra using Wavelet Analysis for Determination of Global Damage.* ISSN 0902-7513 R9517.

PAPER NO. 148: H. U. Köylüoğlu, S. R. K. Nielsen & A. Ş. Çakmak: *Hysteretic MDOF Model to Quantify Damage for TC Shear Frames subject to Earthquakes.* ISSN 1395-7953 R9601.

PAPER NO. 149: P. S. Skjærbæk, S. R. K. Nielsen & A. Ş. Çakmak: *Damage Location of Severely Damaged RC-Structures based on Measured Eigenperiods from a Single Response.* ISSN 0902-7513 R9518.

PAPER NO. 150: S. R. K. Nielsen & H. U. Köylüoğlu: *Path Integration applied to Structural Systems with Uncertain Properties.* ISSN 1395-7953 R9602.

PAPER NO. 151: H. U. Köylüoğlu & S. R. K. Nielsen: *System Dynamics and Modified Cumulant Neglect Closure Schemes.* ISSN 1395-7953 R9603.

PAPER NO. 154: J. D. Sørensen, M. H. Faber, I. B. Kroon: *Optimal Reliability-Based Planning of Experiments for POD Curves.* ISSN 1395-7953 R9542.

PAPER NO. 155: J. D. Sørensen, S. Engelund: *Stochastic Finite Elements in Reliability-Based Structural Optimization.* ISSN 1395-7953 R9543.

PAPER NO. 156: C. Pedersen, P. Thoft-Christensen: *Guidelines for Interactive Reliability-Based Structural Optimization using Quasi-Newton Algorithms.* ISSN 1395-7953 R9615.

PAPER NO. 157: P. Thoft-Christensen, F. M. Jensen, C. R. Middleton, A. Blackmore: *Assessment of the Reliability of Concrete Slab Bridges.* ISSN 1395-7953 R9616.

PAPER NO. 161: S. Engelund, J. D. Sørensen: *Stochastic Models for Chloride-initiated Corrosion in Reinforced Concrete.* ISSN 1395-7953 R9608.

PAPER NO. 165: P. H. Kirkegaard, F. M. Jensen, P. Thoft-Christensen: *Modelling of Surface Ships using Artificial Neural Networks.* ISSN 1593-7953 R9625.

Department of Building Technology and Structural Engineering
Aalborg University, Sohngaardsholmsvej 57, DK 9000 Aalborg
Telephone: +45 98 15 85 22 Telefax: +45 98 14 82 43